Facultad de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN

SAMP+: Una propuesta de comunicación segura de datos astronómicos entre aplicaciones, con retrocompatibilidad

Estudiante: Aitor León Añón
Dirección: José Carlos Dafonte Vázquez
Marco Antonio Álvarez González

A Coruña, septiembre de 2019.

A mis padres, por toda su paciencia conmigo.

Agradecimientos

Me gustaría agradecer en primer lugar a mis padres y a mi hermana por ayudarme en todo lo posible y por el esfuerzo que han tenido que hacer para llegar hasta aquí.

A mis tutores del proyecto. A José Carlos Dafonte por darme la oportunidad de realizar este proyecto con él y por su paciencia ante mis desapariciones. A Marco por tanto tiempo y ayuda en todo y no poner nunca una pega cuando he aparecido.

A mis amigos por sus ánimos y su insistencia.

A mis compañeros de trabajo, que me metieron la presión necesaria para decidirme a terminar este proyecto. A mis compañeros de facultad por todos los buenos momentos.

Muchas gracias a todos.

Resumen

En la actualidad, son cada vez más los proyectos que tienen la necesidad de tratar con grandes cantidades de datos, los cuales tienen que ser puestos a disposición de la comunidad de forma sencilla y accesible para el usuario. Este es el caso del proyecto de la Misión Espacial Gaia de la Agencia Espacial Europea (ESA), donde se pretende crear un catálogo de mil millones de estrellas y, por medio de diversas herramientas, analizar todos sus datos asociados.

Debido al tamaño de este proyecto, cientos de científicos de varios países están colaborando mediante el desarrollo de técnicas, algoritmos y herramientas tanto para la obtención de estos datos, así como su estudio y la posterior divulgación de la información obtenida. Con esta idea central, han sido y están siendo desarrolladas diferentes aplicaciones que permitirán analizar toda esta información desde diferentes perspectivas.

Debido al número de aplicaciones que se han desarrollado para él y a la gran cantidad de datos diferentes que utilizan cada una de ellas, se requiere de un protocolo de mensajería que permita a las herramientas software de astronomía operar y comunicarse entre ellas, este protocolo se denomina SAMP (Simple Application Messaging Protocol) y ha sido desarrollado por la International Virtual Observatory Alliance. Actualmente este protocolo solo soporta comunicaciones entre aplicaciones y herramientas dentro de una misma máquina. Sin embargo, si son mas de una las personas que trabajan simultáneamente y quieren compartir datos, estas comunicaciones se quedan cortas.

Por lo tanto, el objetivo de este proyecto consistirá en, a través del protocolo SAMP, crear un sistema de comunicación remota y segura entre las aplicaciones de la misión Gaia. Para que las comunicaciones sean seguras se implementará un sistema de login mediante LDAP y se cifrarán los mensajes intercambiados entre las aplicaciones mediante SSL.

Abstract

Nowadays, there are more and more projects that have the need to deal with large amount of data, which have to be made available to the community in a simply and accessible way for the user. This is the case of the Gaia Space Mission project of the European Space Agency (ESA), which aims to create a catalog of a billion stars, and analyze all its associated data through different tools. Due to the size of this project, hundreds of scientists from different countries are working together through the development of techniques, algorithms and tools both to obtain these data, as well as their study and the dissemination of the obtained information. With this central idea, different applications have been and are being developed that will allow to analyze all this information from different perspectives.

Because of the magnitude of the Gaia project, the number of applications that have been developed and the huge amount of different data that are used by each one of them, it is needed a message protocol that allows the astronomy software tools to operate and to communicate between them, this protocol is SAMP (Simple Application Messaging Protocol) and it has been developed by the International Virtual Observatory Alliance. Actually, this protocol only supports communications between applications and tools in the same machine. However, if more than one person are working at the same time and they want to share data, these communications fall short.

Therefore, the objective of this project will consist of, through the SAMP protocol, create a remote and secure communication system between applications of the Gaia mission. An LDAP login system will be implemented in order to secure the communications between applications, and the exchanged messages will be encrypted using SSL.

Palabras clave:

- Misión Espacial Gaia
- Agencia Espacial Europea
- Simple Application Messaging Protocol (SAMP)
- Sistema Cliente-Servidor
- LDAP
- SSL
- Socket
- Hub

Keywords:

- Gaia Space Mission
- European Space Agency
- Simple Application Messaging Protocol (SAMP)
- Client-Server System
- LDAP
- SSL
- Socket
- Hub

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este primer capítulo se realizará una presentación del proyecto que describe el documento, sus motivaciones, una especificación de los objetivos que se quieren conseguir y la estructuración de la memoria.

1.1 Motivación

Hoy en día las aplicaciones manejan cantidades de datos cada vez más grandes y es por esto que la necesidad de poder procesar todos estos datos se ve incrementada, de igual manera resulta sumamente importante que toda esta información sea presentada al usuario de una manera que le sea útil. Para esto es necesario disponer de sistemas capaces de aprovechar al máximo las capacidades de cómputo disponibles para procesar, analizar y representar la información disponible.

En el campo de la astronomía, continuamente se desarrollan nuevas aplicaciones y nuevas funciones para manejar estas grandes cantidades de datos. Del mismo modo que este crecimiento es un gran beneficio para los astrónomos, surgen ciertas limitaciones. De manera inevitable las funciones de estas herramientas se solapan debido a que cada usuario exige capacidades de otras en sus herramientas favoritas. De este modo, al mismo tiempo que las herramientas se vuelven más poderosas, se incrementa su complejidad y su interoperabilidad se complica.

Una manera alternativa y complementaria de aproximarse a este problema es la colaboración entre las herramientas, especializándose cada una de ellas en una tarea particular. Esta aproximación permite a los equipos astronómicos ensamblar diferentes herramientas en función de sus propias necesidades.

Con el fin de que estas herramientas se puedan ensamblar entre sí y se complementen, surge SAMP[1, 2] (Simple Application Messaging Protocol) desarrollado por IVOA [3] (International Virtual Observatory Alliance). SAMP es el estándar para compartir datos en aplica-

ciones de astronomía compatibles con VO y surge de los trabajos de interoperabilidad de los equipos Aladin [4, 5] y VisIVO.

En los tiempos que corren, el uso generalizado de internet propicia la posibilidad de poder conectar distintos ordenadores con la finalidad de poder trabajar simultáneamente con los mismo datos y compartir al instante los resultados de los trabajos realizados por los diferentes usuarios y herramientas. El protocolo SAMP permite a todas las aplicaciones que lo utilizan, a través de un Hub central, interoperar y comunicarse entre ellas, facilitando la diferenciación entre el trabajo con los datos disponibles y la puesta en común de los mismos entre las diferentes herramientas que trabajan en un mismo proyecto. De esta forma, a partir de una aplicación y el resultado de su trabajo con los datos, éstos puedan enviarse a través de SAMP de una aplicación a otra, para su posterior uso. Pese a todo, SAMP tiene ciertas limitaciones, que se pretenden salvar con el trabajo que se va a realizar en este proyecto mediante el desarrollo de un nuevo estándar, JSAMP+.

Este proyecto forma parte de los trabajos realizados por el grupo L(IA)2 de la Facultad de Informática de la Universidade da Coruña como parte de su contribución a la misión Gaia [6, 7].

Gaia es un proyecto de la ESA [8] (Agencia Espacial Europea) cuyo objetivo es el de examinar más de mil millones de estrellas en nuestra galaxia, así como objetos extra-galácticos (otras galaxias, cuásares, etc.). El procesamiento de esta gran cantidad de datos permitirá a los astrónomos responder a algunas de las preguntas acerca de la formación y evolución de nuestra galaxia, y construir un mapa 3D preciso de los objetos celestes de ella. Debido a la gigantesca cantidad de información que se maneja es necesaria la colaboración de gran cantidad de científicos de varios países mediante el desarrollo de técnicas tanto para la obtención de estos datos como para su estudio y divulgación, de la misma forma se necesitan herramientas que pongan a disposición de la comunidad todo este conocimiento. Y es en este ámbito donde adquiere su importancia SAMP y en el caso de nuestro proyecto JSAMP+.

1.2 Objetivos

Entre las limitaciones de SAMP está que sólo permite conexiones entre las aplicaciones de un mismo ordenador y debido a ello las conexiones que utiliza entre las diferentes aplicaciones no disponen de seguridad alguna. Por lo tanto, el objetivo principal de este proyecto será el de mejorar el estándar SAMP proporcionando un nivel de seguridad en las comunicaciones entre las aplicaciones a través de la creación de un nuevo estándar escrito en JAVA [9], al que hemos bautizado como JSAMP+.

Como objetivos concretos, tenemos:

- Crear un sistema de autenticación mediante el uso de LDAP, que solicite usuario y contraseña a las aplicaciones que se conecten al Hub central.
- Permitir conexiones remotas al Hub, ya que el estado actual del protocolo SAMP solo permite la conexión de aplicaciones locales.
- Cifrar las conexiones y los intercambios de mensajes y de datos entre las diferentes aplicaciones y el Hub, mediante un servidor HTTPS.
- Mantener la compatibilidad con el estándar SAMP clásico, ya que todas las aplicaciones que existen todavía lo utilizan.

Dentro del desarrollo de la aplicación, se añadirá la capacidad de que las comunicaciones soporten varios formatos de mensajes, permitiendo así que los usuarios puedan enviar datos a otras aplicaciones no solo según su tipo de datos, sino también a las diferentes aplicaciones en las que un usuario se encuentra logueado.

Por último, es necesario destacar que a la hora de proceder con el desarrollo de este proyecto, se partirá del trabajo realizado por Mark Taylor con JSAMP [10]. JSAMP es un proyecto Maven [11] formado por una librería JAVA y herramientas para usar SAMP. Comprime un cliente, una herramienta de línea de comandos y un Hub. Es éste Hub a partir del cual comenzaremos el desarrollo del proyecto.

1.3 Organización de la memoria

Para facilitar la lectura y seguimiento del trabajo realizado en el proyecto, la memoria se estructura en siete capítulos que explican de una manera práctica y ordenada el trabajo realizado.

- ***Capítulo 1: Introducción.*** Se realiza una descripción de cuales son los motivos que llevaron al desarrollo del proyecto y los objetivos que se quieren alcanzar.
- ***Capítulo 2: Contextualización.*** En este capítulo se realiza una breve explicación del proyecto Gaia, del funcionamiento de SAMP y de las tecnologías con las que se pretende desarrollar este proyecto.
- ***Capítulo 3: Estado del Arte.*** En este apartado se realiza una breve explicación de la implementación actual de SAMP y de cual es su funcionamiento con las herramientas actuales.
- ***Capítulo 4: Estudio de viabilidad.*** Se comentará la metodología elegida para el desarrollo del proyecto, su planificación y seguimiento, así como los factores de riesgo asociados.
- ***Capítulo 5: Desarrollo.*** En este apartado se comentará todo el proceso de desarrollo del proyecto.
- ***Capítulo 6: Conclusiones.*** El objetivo de este apartado será resumir las conclusiones que se han alcanzado sobre el proyecto y revisar el cumplimiento de los objetivos iniciales.
- ***Capítulo 7 Líneas futuras.*** Se sugerirán los siguientes pasos que se podrían dar con el fin de establecer el nuevo estándar JSAMP+.

Capítulo 2

Contextualización

Antes de explicar el desarrollo del presente proyecto es necesario poner en contexto el ámbito del mismo. Para ello es necesario familiarizarse con el proyecto Gaia y con el estándar SAMP. A mayores se explicarán las diferentes tecnologías utilizadas para realizar el trabajo.

2.1 El proyecto Gaia

En septiembre de 2013, la Agencia Espacial Europea (ESA) puso en órbita el satélite de la misión espacial Gaia (Figura 2.1). Es una misión sucesora de la misión Hipparcos [12], lanzada en 1989 e incluida junto a Gaia dentro del contexto del programa científico ESA Horizon 2000+. Se encuentra en una órbita de Lissajous [13] en el punto L2 de Lagrange alrededor del sistema Sol-Tierra a una distancia de 1,5 millones de kilómetros de la Tierra en la dirección anti-Sol, co-girando con la Tierra en su órbita. Gaia examinará y recogerá datos de más de mil millones de estrellas en nuestra galaxia así como objetos extra-galácticos (otras galaxias, cuásares). Este gran número de datos estelares permitirá a los astrónomos responder a algunas de las preguntas acerca de la formación y evolución de nuestra galaxia, y construir un mapa 3D preciso de los objetos celestes de ella. De este modo, el objetivo de la misión es crear un catálogo astronómico tridimensional de nuestra galaxia, la Vía Láctea, así como recoger información de movimientos, composición, formación y evolución de la misma.

Gaia proporcionará mediciones astrométricas para determinar las posiciones, distancias y movimientos de las estrellas; fotométricas, tales como luminosidad, temperatura y composición química de cada estrella, y medidas de velocidad radial; todo ello con la precisión necesaria para producir un censo estereoscópico y cinemático de aproximadamente mil millones de estrellas en nuestra galaxia.

La sonda monitorizará cada estrella alrededor de 70 veces durante un periodo que estaba programado inicialmente para 5 años, pero que se ha extendido otro año más, precisando

posiciones, distancias, movimientos y cambios de luminosidad. Con ello se espera que se descubran miles de nuevos objetos celestes, como planetas extra solares o enanas marrones; y se observarán miles de asteroides dentro de nuestro Sistema Solar. Del mismo modo proporcionará nuevas pruebas de la Teoría General de la Relatividad de Albert Einstein.

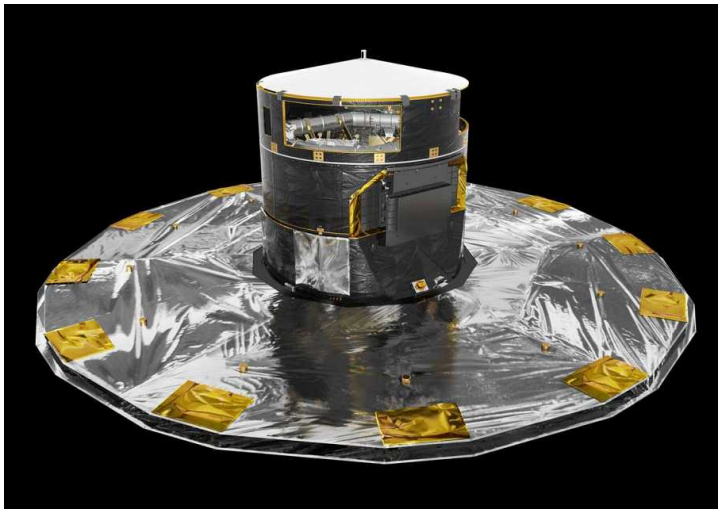


Figura 2.1: Sonda Espacial Gaia. Imagen: ESA/ATG

En general, las observaciones y tránsitos de estos años serán publicados con el catálogo final. Sin embargo, cuando se publican datos sobre estrellas y sus variaciones, también se pueden publicar subconjuntos relevantes de datos de ese periodo de tiempo. Cuando la publicación de los datos es de inmediato interés científico, se realiza sobre una base individual a través de alertas científicas.

La naturaleza de la misión la lleva a la adquisición de una enorme cantidad de datos complejos y extremadamente precisos, representando las múltiples observaciones de mil millones de objetos diferentes. El desafío de Gaia (procesando telemetría satelitar para convertirla en productos científicamente válidos) es una tarea enorme en términos de experiencia, esfuerzo y poder de computación dedicado. Para procesar todos los datos de Gaia se creó un gran equipo de científicos y desarrolladores pan-europeos conocidos como DPAC [14] (Data Processing and Analysis Consortium). Conformado por más de 20 países, este consorcio aúna las habilidades y experiencia de todo el continente, reflejando la naturaleza internacional y cooperativa de la ESA. El DPAC existe desde 2006 y tiene la tarea de desarrollar algoritmos de procesamiento de datos, el correspondiente software y la infraestructura IT de Gaia. Las responsabilidades del consorcio son :

- Preparación de los algoritmos de análisis de datos para reducirlos en un framework de procesamiento integrado y coherente.

- Generación y suministro de datos simulados para soportar el diseño, desarrollo y prueba del sistema de procesamiento de datos.

- Diseño, desarrollo, búsqueda y operación de todos los aspectos de procesado hardware y software necesarios.

- Diseño, desarrollo, y operación de la base de datos y el archivo de Gaia.

Este consorcio está dividido en nueve unidades especializadas conocidas como Unidades de Coordinación o CUs, estando cada unidad asignada al procesamiento de un único conjunto de datos. Los CUs están distribuidos por múltiples países y están apoyados por los seis centros de Procesamiento de datos, o DPCs (Figura 2.2). Éstos son los centros donde está disponible el hardware de procesamiento. La Figura 2.3 muestra los diferentes roles de los CUs y DPCs.

Figura 2.2: Centros de procesado de datos. Imagen: ESA/Gaia/DPAC

El grupo L(IA)2 de la Facultad de Informática, forma parte desde hace años de las Unidades de Coordinación CU-8 y CU-9, encargadas de la caracterización Astrofísica y el Acceso al Archivo y Catálogo respectivamente.

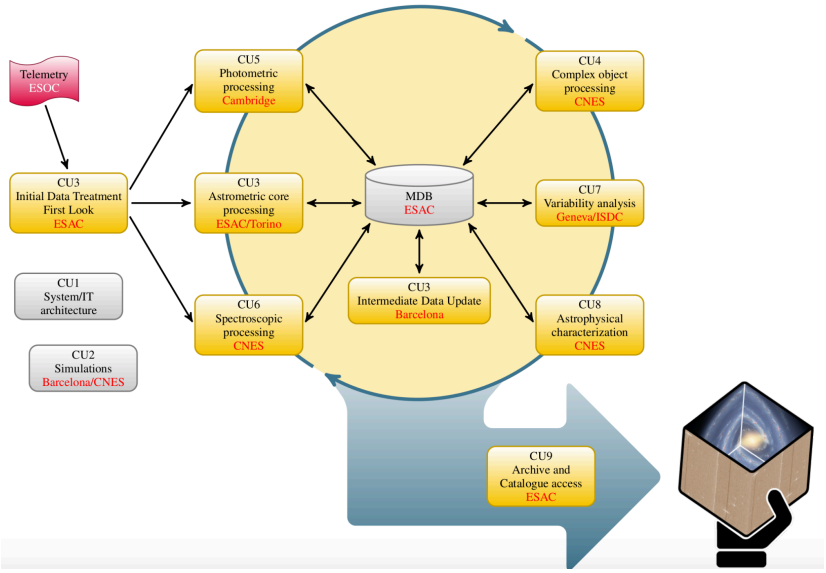


Figura 2.3: CUs y sus roles. Imagen: ESA/Gaia/DPAC

2.1.1 Catálogos astronómicos

Un catálogo astronómico es un listado o tabla de objetos astronómicos agrupados por alguna característica común. Estas características pueden ser morfología, origen, tipo, descubrimiento, etc.

Existen diferentes catálogos que se han creado en proyectos anteriores de los que vamos a destacar el Catálogo Hipparcos (misión predecesora a Gaia), que contiene datos de 118.000 estrellas, y el Catálogo Tycho-2, que contiene datos de 2.500.000 estrellas y que es usado en la misión Gaia para establecer comparaciones.

La misión Gaia, creará un nuevo catálogo de mil millones de estrellas que será utilizado posteriormente para el estudio del espacio exterior. Como en este proyecto se necesita poder ubicar dichos objetos en el espacio, es necesario explicar cómo se realiza esta tarea a través de **coordenadas astronómicas**.

Las **coordenadas astronómicas** son el conjunto de valores que dan la posición de un objeto en la esfera celeste, en función de un sistema de referencia. Para localizar un punto en la bóveda terrestre se utilizan declinación y ascensión recta. La **declinación** es el ángulo que forma un astro con el ecuador celeste. La **ascensión recta** se mide en horas a partir del punto Vernal (posición del Sol en el equinocio de Primavera).

2.1.2 Publicaciones

En general, las observaciones y tránsitos individuales se publicarán solo con el catálogo final. Sin embargo, cuando se publican datos variables de estrellas, también se pueden publicar conjuntos de datos relevantes. Adicionalmente, cuando esta publicación es de interés científico inmediato, se realizará de forma individual a través de alertas científicas.

Tras el procesamiento de los datos y la producción del catálogo para la publicación, se ejecuta un ejercicio de validación. Existen dos publicaciones realizadas de la misión Gaia y se espera otra antes de la publicación final:

- DR1 [15]: 14 de septiembre de 2016.
- DR2 [16]: 25 de abril de 2018.
- EDR3 y DR3 [17]: Publicación dividida en dos partes, una en el tercer cuatrimestre de 2020 y otra en la segunda mitad de 2021.

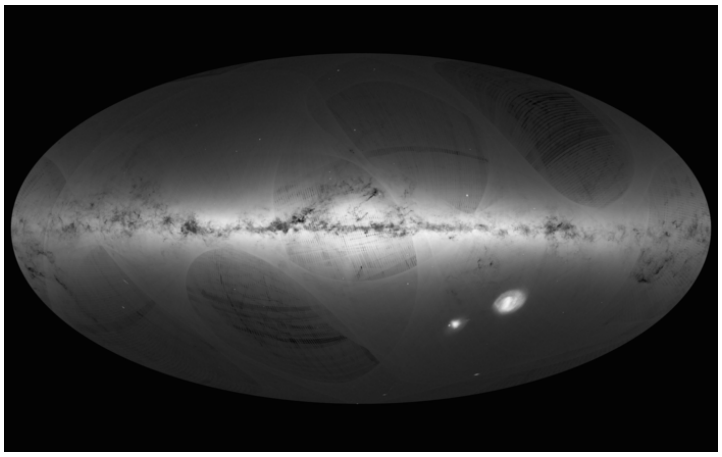


Figura 2.4: DR1: Primer mapa del cielo de Gaia . Imagen: ESA/Gaia/DPAC

DR1

Esta primera publicación consiste en:

- Posiciones y magnitudes Gaia para 1.100 millones de estrellas.
- Posiciones y movimientos de las estrellas usando la Solución Astronómica Tycho-Gaia (TGAS).
- Posiciones y magnitudes de cuásares.

- Curvas y características de 3000 estrellas variables.
- Coincidencias entre las fuentes de Gaia y proyectos anteriores similares.

Además en esta publicación se crea un primer mapa en blanco y negro del cielo (Figura 2.4).

DR2

La segunda publicación consiste en:

- Posiciones y magnitudes Gaia para cerca de 1.700 millones de estrellas.
- Velocidades radiales para mas de siete millones de estrellas.
- Parámetros astrofísicos tales como temperatura de la superficie, radio y luminosidad.
- Posición y época de observación de 14.099 objetos del Sistema Solar.
- Clasificación para más de 550.000 estrellas.
- Coincidencias entre las fuentes de Gaia y proyectos anteriores.

En esta segunda publicación se consigue una imagen en color del mapa del cielo, hecho por el satélite (Figura 2.5).

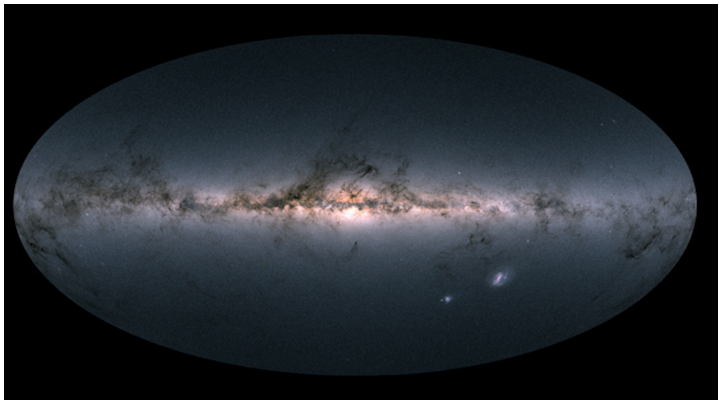


Figura 2.5: DR2: Mapa en color del cielo de Gaia. Imagen: ESA/Gaia/DPAC.

EDR3 y DR3

El catálogo Gaia EDR3 consistirá en:

- Astrometría y fotometría mejorada.
- Resultados sobre cuásares y objetos extendidos.

La segunda parte del catálogo (DR3) consistirá en:

- Clasificación de objetos y parámetros de astrofísica.
- Datos y clasificaciones de estrellas.
- Resultados del sistema solar.

Publicación final

Se espera que la publicación final de la misión Gaia sea en 2022, y que contenga el catálogo completo de todos los objetos de la Vía Láctea. Este catálogo consistirá en:

- Catálogos astronómicos, fotométricos y de velocidad radial completos.
- Clasificación y múltiples parámetros astrofísicos de las estrellas.
- Lista de exo-planetas.

2.2 SAMP

SAMP [1] (Simple Application Messaging Protocol) es un protocolo de mensajería que permite a herramientas de software de astronomía operar entre ellas y comunicarse. Los miembros de IVOA (Internation Virtual Observatory Alliance) han reconocido que crear una herramienta monolítica que intente ajustarse completamente a los requisitos de todos los usuarios es inviable, y es mejor usar nuestros recursos para permitir que las diferentes herramientas puedan trabajar juntas. Para ello, es necesario definir un formato de archivos común para el intercambio de datos entre todas las aplicaciones y un sistema de mensajería que permita a las aplicaciones compartir datos y tomar ventaja de las funcionalidades de cada una de ellas. SAMP es un descendiente de protocolo PLASTIC [18] que creció de la interoperabilidad de los trabajos de los equipos de Aladin [5] y VisIVO.

SAMP es un framework abstracto para comunicación débilmente acoplada, asíncrona, similar a RPC [19] y/o basada en eventos; está basado en un servicio central que proporciona intermediación de mensajes de publicación/suscripción multidireccional.

El estándar SAMP no es dependiente de otros estándares VO (Virtual Observatory), sino que proporciona una valiosa unión entre las aplicaciones de nivel de usuario que realizan diferentes tareas relacionadas con VO, y por lo tanto contribuye a la integración de su funcionalidad desde el punto de vista del usuario. La Figura 2.6 ilustra SAMP en el contexto de la Arquitectura IVOA. La mayoría de las herramientas existentes que operan en la capa de usuario de esta arquitectura proporcionan interoperabilidad SAMP.

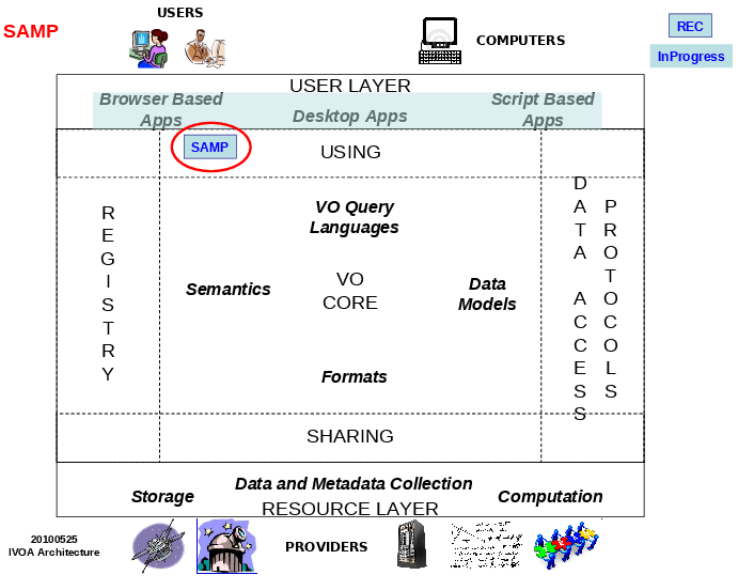


Figura 2.6: Diagrama de la arquitectura IVOA. El protocolo SAMP aparece en la región "Using". Imagen: IVOA/SAMP

La semántica de los mensajes que acepta el protocolo está definida por contratos conocidos como MTypes (Message-Types), que están definidos por acuerdos con los desarrolladores. La lista de los MTypes usados pueden ser encontrados en http://www. ivoa.net/samp/. Los mensajes que se envían y los metadatos que se almacenan de los diferentes clientes están en formato JSON [20].

Actualmente no tiene soporte para transacciones, seguridad ni para garantizar la integridad ni la entrega de los mensajes; aunque la arquitectura de SAMP permite que esas capacidades se pueden añadir en un futuro.

SAMP tiene una arquitectura basada en un Hub, que es un servicio utilizado para enrutar los mensajes entre los clientes. Estos clientes se basan en un Application Discovery, que consiste en un proceso a través del cual las aplicaciones conectadas son identificadas e inventariadas. La arquitectura de SAMP hace más sencillo este proceso ya que cada cliente solo necesita conocer el Hub y los servicios proporcionados por el mismo se encargan de simplificar las acciones del cliente.

El proceso básico de funcionamiento de un cliente SAMP es el siguiente:

1. El cliente determina si existe un Hub corriendo usando el mecanismo de descubrimiento de Hub apropiado; si lo hay se registra con él.

2. El Hub se encarga de almacenar los metadatos del cliente, asi como los MTypes a los que se suscribe el cliente.

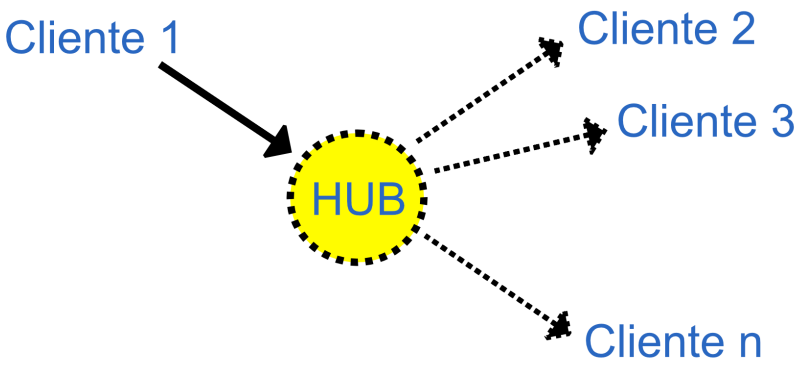


Figura 2.7: Arquitectura del Hub de SAMP

3. El cliente pregunta al Hub por los metadatos del resto de clientes; y envía y/o recibe mensaje a/desde otros clientes a través del hub.

4. El cliente se desconecta del Hub.

El funcionamiento del Hub es:

1. Detectar si ya existe algún Hub corriendo mediante un procedimiento denominado Hub Discovery. Si ya existe un Hub, se detiene el nuevo, si no, se inicia.
2. Esperar registro de clientes. Cuando un cliente se registra correctamente, se le asigna un ID, se añade a la tabla de clientes registrados y se anuncia el registro del nuevo cliente a las demás aplicaciones ya conectadas al hub. El Hub tiene disponibles los metadatos de los clientes y, cuando se produce algún cambio, lo anuncia a través de una difusión múltiple a todos los clientes.
3. Cuando un cliente actualiza sus MTypes, se realiza una difusión de los mismos a todos los clientes conectados al hub.
4. Cuando el Hub recibe un mensaje, lo envía a los clientes suscritos al MType del mensaje, excepto al cliente emisor del mensaje.
5. Espera a que los clientes se desconecten y envía por broadcast un mensaje de aviso al resto de clientes registrados.
6. Cuando el Hub se va a apagar envía un mensaje por broadcast a todos los clientes registrados.

2.2.1 Patrones de Mensajes

Los mensajes pueden ser enviados mediante 3 patrones, difiriendo en cuándo y cómo la respuesta es enviada al emisor:

- Notificación.
- Llamada/Respuesta Asíncrona.
- Llamada/Respuesta Síncrona.

Si el emisor espera recibir algún dato como resultado de la respuesta del receptor, o si desea conocer el momento en que se completa el proceso de envío, debe utilizar alguna de las variantes de Llamada/Respuesta. Si en cambio no tiene interés en recibir ninguna respuesta utilizará la variante de Notificación. El Hub debe procesar del mismo modo todos los mensajes, sin tener en cuenta cual de estos 3 patrones sea el utilizado.

2.2.2 Consideraciones de seguridad

SAMP permite la comunicación interprocesal incluyendo la capacidad de que un cliente cause la ejecución de código de otro cliente. Esto genera la posibilidad de que un cliente

sin privilegios ejecute acciones privilegiadas en virtud de la interoperabilidad permitida por SAMP. En la practica esto depende de la identidad de los clientes y de la semántica de los mensajes. Se recomienda no usar MTypes que puedan provocar la ejecución de código arbitrario, o declarar metadatos que revelen información sensitiva.

Además, el Hub SAMP solo permite registrar clientes desde el localhost. Y solo permite añadir manualmente hosts conocidos a la lista de hosts permitidos para su conexión. Abrir el acceso a todo el mundo abriría la posibilidad de ataques DOS y quizás ataques PHISING en los cuales el usuario queda expuesto a peticiones SAMP de registro no deseadas.

El protocolo SAMP recomienda también que cada vez que se registre un nuevo cliente contra el hub, solicite una confirmación al usuario del mismo, indicando el origen del cliente y el nombre del cliente.

2.2.3 Envío y recepción de mensajes

Según los 3 patrones de mensajes, para un MType dado hay un patrón típicamente usado, pero no hay nada que impida que pueda usar otro patrón.

Desde el punto de vista del emisor hay 3 modos de enviar el mensaje y desde el punto de vista del receptor hay 2 modos de recibirlo.

- Notificación.

  La comunicación va en una sola dirección:

  1. El emisor envía el mensaje al Hub para que lo entregue a uno o más receptores.
  2. El Hub envía el mensaje a los receptores.
  3. No se espera respuesta del receptor.

Figura 2.8: Patrón notificación

- Llamada/Respuesta Asíncrona.

  Usa etiquetas de mensajes e identificadores de mensajes para agrupar mensajes y respuestas:

  1. El emisor envía el mensaje al Hub para que lo entregue a uno o mas receptores, junto con una etiqueta. Como respuesta recibe un identificador de mensaje (msg-id).

2. El Hub envía el mensaje a los receptores y el msg-id.
3. Cada receptor procesa el mensaje y envía una respuesta al Hub con el msg-id.
4. Usando un procedimiento de llamada de vuelta el Hub envía la respuesta de vuelta al original junto con el msg-tag.

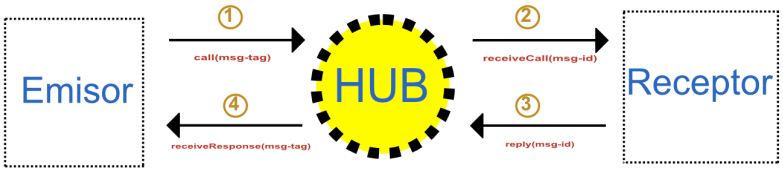


Figura 2.9: Patrón Llamada/Respuesta Asíncrona.

- Llamada/Respuesta Síncrona.

  Un método síncrono es proporcionado por el Hub, principalmente para los ambientes en los que la asincronía pueda ser un problema. El Hub proporciona sensación de sincronía al emisor, interactuando con el receptor de la misma manera que si fuera asíncrona.

  1. El emisor envía el mensaje al Hub para que lo entregue a uno o mas receptores, opcinalmente especifica un máximo de tiempo de espera. La llamada se bloquea hasta que la respuesta esté disponible.
  2. El Hub envía el mensaje al receptor, junto con el msg-id.
  3. El receptor procesa el mensaje y envía la respuesta junto con el msg-id.
  4. El Hub envía la respuesta como valor resultado de la llamada original hecha por el emisor.

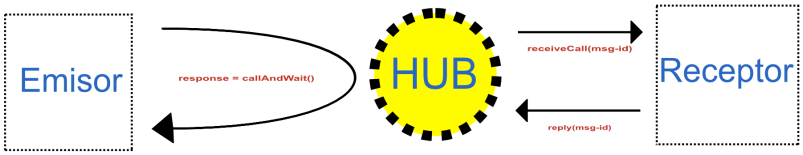


Figura 2.10: Patrón Llamada/Respuesta Síncrona.

2.2.4 JSAMP

JSAMP es un toolkit de Java para usar con SAMP. Está desarrollado por Mark Taylor [2] trabajando con el Grupo Astrofísico de la Universidad de Bristol. Proporciona los siguientes componentes:

- Una implementación de un Hub SAMP.
- Un grupo de clases que pueden ser usadas para implementar capacidades en aplicaciones clientes.
- Utilidades que pueden ser usadas por desarrolladores de clientes y hubs, o de usuarios de SAMP.
- Componentes gráficos para el uso básico de las funcionalidades superiores.
- Componente bridge, que permite a clientes de diferentes hubs a hablar entre si.

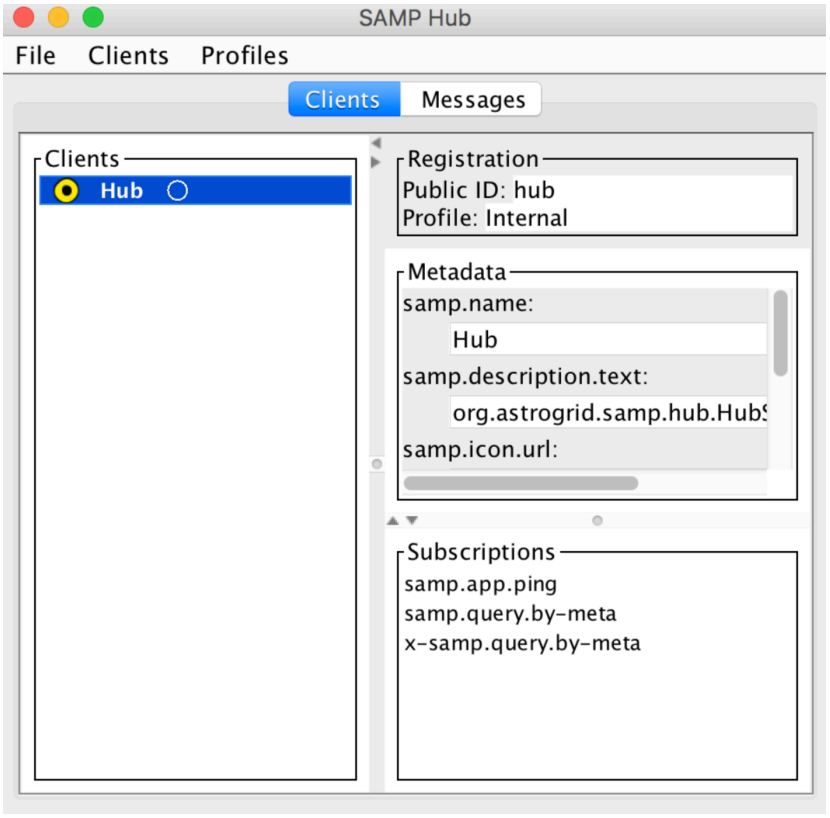


Figura 2.11: JSAMP Hub. Pestaña Clientes.

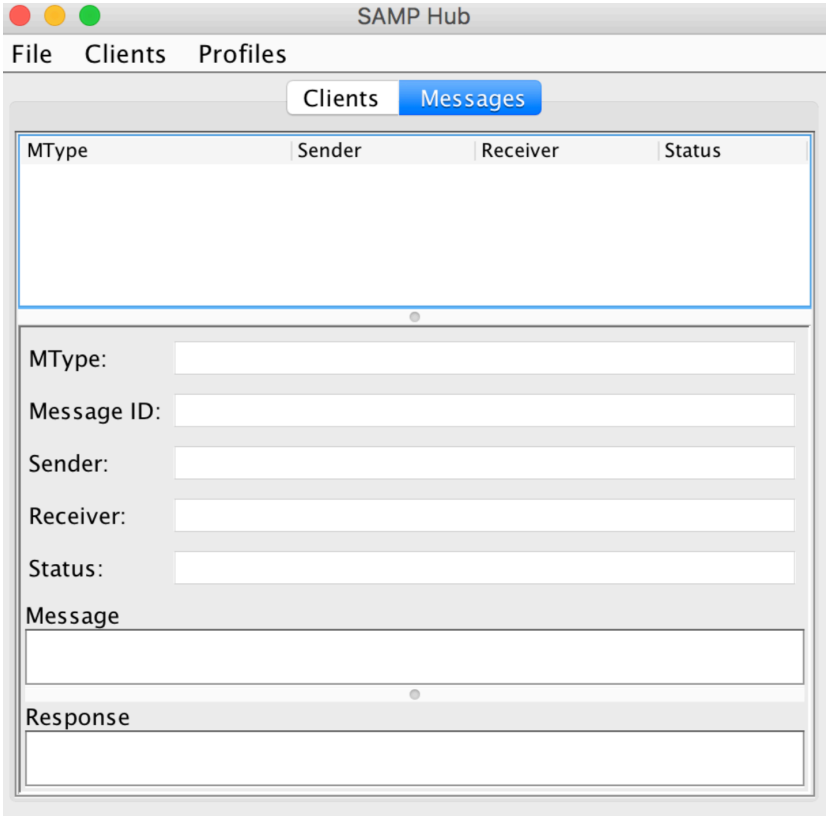


Figura 2.12: JSAMP Hub. Pestaña Mensajes.

2.3 LDAP

El protocolo ligero de acceso a directorios (LDAP) [21] es un protocolo estándar de aplicación, abierto, para acceder y mantener servicios de información de directorio distribuidos a través de una red IP. Los servicios de directorio desempeñan un papel importante en el desarrollo de aplicaciones de intranet e Internet al permitir el intercambio de información sobre usuarios, sistemas, redes, servicios y aplicaciones en toda la red. Como ejemplos, los servicios de directorio pueden proporcionar cualquier conjunto organizado de registros, a menudo con una estructura jerárquica, como un directorio de correo electrónico corporativo.

LDAP está especificado en una serie de publicaciones RFC (Request for Comments) que pertenecen al Internet Engineering Task Force (IETF), utilizando el lenguaje de descripción ASN. La última especificación es la Version 3.

Un uso común de LDAP es proporcionar un lugar para almacenar nombres de usuario y contraseñas. Esto permite que muchas aplicaciones y servicios diferentes se conecten al servidor LDAP para validar usuarios.

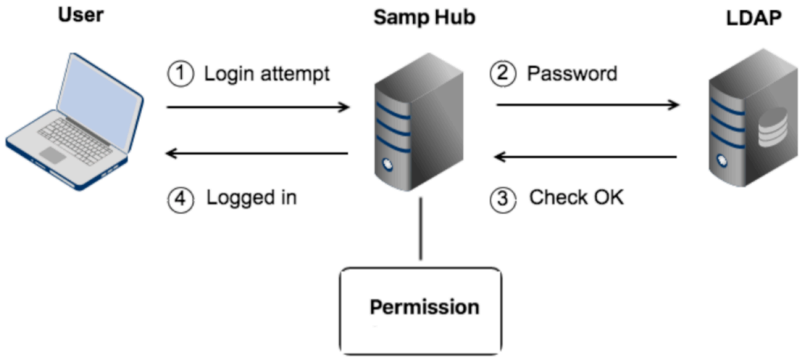


Figura 2.13: Uso de LDAP para Autenticacion en SAMP hub.

2.4 SOCKETs

Un socket [22] es un extremo de un link de comunicaciones de doble sentido entre 2 programas corriendo en una red. Está unido a un número de puerto de manera que la capa TCP puede identificar la aplicación a la cual los datos a enviar están destinados.

Normalmente, un servidor corre en un ordenador especifico y tiene un socket unido a un puerto específico. El servidor solo espera, escuchando al socket, a que un cliente realice una petición de conexión.

El cliente sabe el nombre de la máquina en la que se encuentra el servidor y el número del puerto en el que se encuentra escuchando. Para realizar la conexión, el cliente debe realizar

una petición de conexión al socket. Es necesario que el cliente se identifique con el servidor, y se enlace a un número de puerto para toda la conexión.

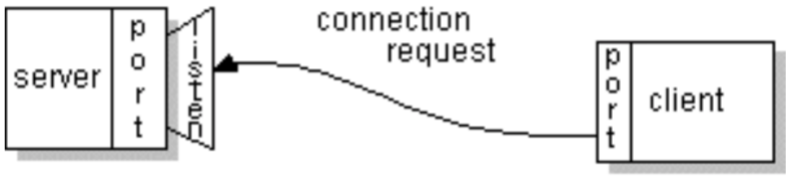


Figura 2.14: Sockets. Petición de conexión del cliente.

Si todo va bien, el servidor acepta la conexión. Tras esto el servidor obtiene un nuevo socket vinculado al mismo puerto local. Se necesita un nuevo socket para que pueda seguir escuchando el socket original las peticiones de conexiones mientras atiende al cliente conectado.

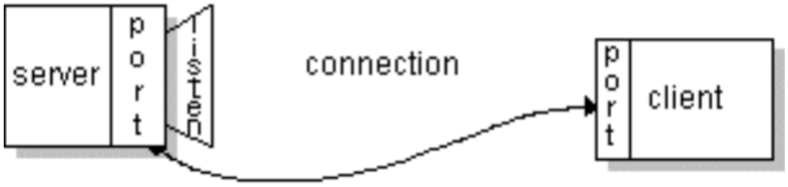


Figura 2.15: Sockets. Conexión establecida entre servidor-cliente.

En el lado del cliente, si la conexión es aceptada, se crea un socket correctamente, y el cliente puede usar el socket para comunicarse con el servidor.

2.4.1 Sockets en java.net

La clase socket [23] del paquete java.net de la plataforma Java implementa un lado de una conexión de doble sentido entre un programa Java y otro programa en la red. La clase Socket se sitúa sobre una implementación dependiente de plataforma, ocultando los detalles de cualquier sistema particular al programa Java.

Adicionalmente, se incluye también la clase ServerSocket, que implementa un socket cuyos servicios pueden ser usados para escuchar y aceptar conexiones de clientes.

2.5 SSL

SSL (Secure Socket Layer) [22] es una tecnología de seguridad estándar usada para el establecimiento de un enlace encriptado entre un servidor y un navegador web. Este enlace asegura que todas las comunicaciones se mantienen privadas. También recibe el nombre de TLS (Transport Layer Security). Millones de sitios web utilizan encriptación SSL para asegurar las conexiones y mantener los datos a salvo.

Para poder crear una conexión SSL, un servidor requiere un certificado SSL. Para crear el certificado, se solicitan una serie de preguntas sobre la identidad del sitio y de la empresa. El servidor crea entonces 2 claves criptográficas: una privada y otra pública.

La clave pública no necesita ser secreta y se coloca en un Certificate Signing Request (CSR)(un archivo que contiene los datos del usuario). A continuación, se envía el CSR. Durante el proceso de solicitud de Certificado SSL, la Autoridad de Certificación validará los datos del usuarioy emitirá un Certificado SSL que contendrá los datos del usuario y permitirá el uso de SSL. El certificado SSL emitido del servidor coincide con su clave privada.

La arquitectura general del protocolo SSL consta de las siguientes capas:

- Protocolo de Registro SSL
- Protocolo de Handshake SSL
- Protocolo de especificación de cifrado de SSL
- Protocolo de alerta SSL

Existen 2 conceptos básicos de SSL:

- Conexión SSL: En SSL los mecanismos de conexión son relaciones P2P(Peer-to- peer).
- Sesión SSL: Asociación cliente-servidor. Creada por el protocolo de Handshake.

2.5.1 Protocolo de Registro SSL

Proporciona 2 servicios principalmente:

1. Integridad
2. Confidencialidad

El protocolo de registro toma la información de la aplicación y la fragmenta en bloques. Entonces los comprime y añade la MAC para encriptar los datos. Finalmente se añade la cabecera y es transmitida al segmento TCP. Luego los datos se fragmentan en bloques de 214 bytes. Posteriormente se produce la compresión, aunque es opcional. Posteriormente, se calcula la

MAC en el que se utiliza la clave secreta compartida. Tras esto se produce la encriptación con el algoritmo elegido.

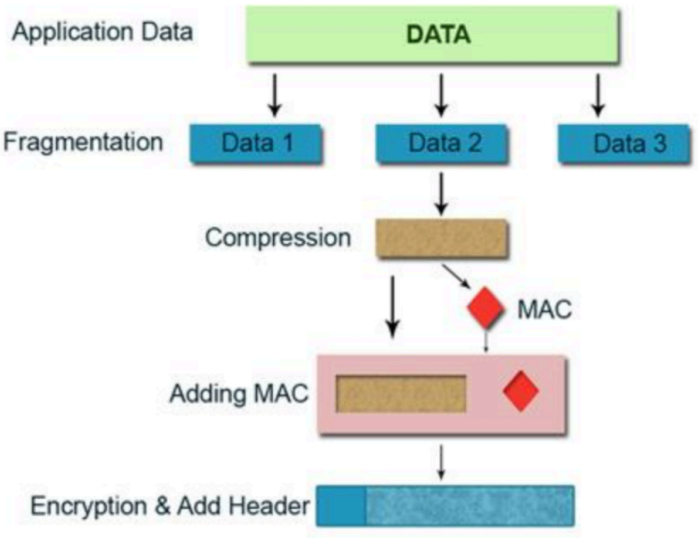


Figura 2.16: Funcionamiento del protocolo de Registro SSL.

El paso final es añadir la información de cabecera al contenido cifrado.

2.5.2 Protocolo de Especificación de Cifrado SSL

Utiliza el protocolo de registro. Es un protocolo TLS. Consiste en un único mensaje de un byte. Causa un estado pendiente para copiar copiarlo al estado actual. Actualiza el texto encriptado para poder ser usado en TLS.

2.5.3 Protocolo de alerta SSL

Usado normalmente para enviar y recibir Alertas relacionadas con SSL entre entidades end to end. Estos mensajes de alerta son comprimidos y encriptados.

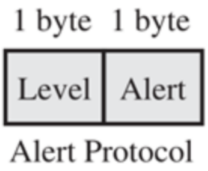


Figura 2.17: Mensaje del protocolo de alerta SSL.

Cada mensaje usa 2 bytes. El byte de nivel es un valor de advertencia y el byte de alerta transmite la severidad de los datos útiles. Si el nivel es fatal, SSL inicia directamente la

conexión. Si hay más conexiones al mismo tiempo, éstas deben continuar. El byte de alerta contiene el código para producir una alerta específica. Estas alertas pueden ser:

- Bad-certificate
- Unsupported-certificate
- ...

2.5.4 Protocolo de Handshake SSL

La parte mas importante y complicada del SSL es el handshake. Este protocolo permite a los dos extremos conectarse entre ellos, autenticándose, negociando la encriptación y intercambiando paquetes. Está dividido en 4 fases:

Fase 1. Estableciendo la Conexión y las capacidades para la seguridad

Inicia la conexión lógica entre un cliente y un servidor. También genera capacidades de seguridad que usarán cliente y servidor.

Figura 2.18: Fase 1 del Handshake SSL.

Primero, el cliente envía un mensaje de saludo (client-hello) y espera a que el servidor conteste. El servidor contesta al cliente con otro mensaje de saludo (”server-hello”). Ambos mensajes contienen los mismos parámetros, tales como versión, Id de sesión, mecanismo de cifrado y compresión y números aleatorios iniciales. Tras esto, el servidor genera un campo aleatorio diferente al del cliente.

Fase 2. Intercambio de claves y Autenticación del servidor

El servidor envía un certificado al cliente que va a ser autenticado. Contiene uno o mas cadenas de certificados X.509. Entonces, el servidor envía un mensaje de intercambio de clave si es necesario.

Figura 2.19: Fase de intercambio de claves de servidor.

Fase 3. Intercambio de claves y Autenticación del cliente

Después de que el servidor envíe el certificado al cliente, el cliente verifica si es valido o no. Si todo está bien, el cliente envía mas de un mensaje de vuelta al servidor. Tras esto el cliente envía un mensaje de intercambio de clave.

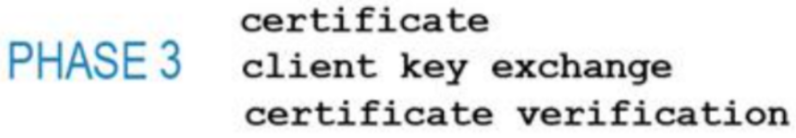


Figura 2.20: Fase de intercambio de clave de cliente.

El cliente envía un mensaje de veracidad del certificado. Este certificado siempre tiene capacidad de firma. Firma un hash basado en el mensaje anterior. Para llevar a cabo la verificación se realiza la firma del certificado.

Fase 4. Fase Final

La cuarta fase crea la conexión segura. El cliente envía un mensaje de especificación de cifrado. El cliente copia el nuevo cifrado en la actual especificación de cifrado y envía un mensaje final. Solo el mensaje final verifica que la autenticación y el intercambio de claves ha sido el correcto, y se puede empezar el intercambio de datos.

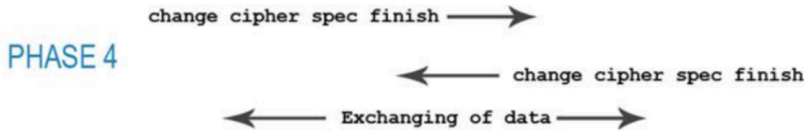


Figura 2.21: Fase Final Handshake.

En la siguiente imagen se muestra un resumen de todo el proceso.

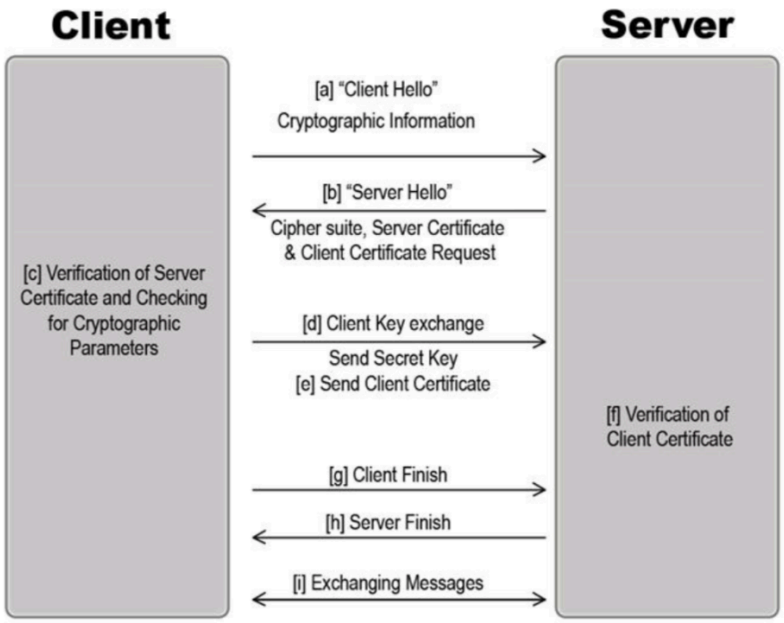


Figura 2.22: Proceso SSL completo.

2.6 JSON

JSON [20] (JavaScript Object Notation) es un formato ligero de intercambio de datos y fácil de procesar. Surge como un intento para definir formatos más ligeros y rápidos que XML a principios de los años 2000 y se caracteriza por reducir el volumen de datos que es necesario transmitir frente a las grandes cantidades de datos que se utilizan en otros estándares como XML. Este formato está basado en 2 estructuras universales de datos:

- Pares nombre/valor. Según el lenguaje en el que se implemente el JSON, estos pares son reconocidos como objetos, estructuras, diccionarios...
- Lista ordenada de valores. Según el lenguaje: array, vector, lista o secuencia.

Actualmente todos los lenguajes de programación tienen soporte para estas estructuras de una o de otra manera, por lo que es lógico que JSON esté basado en ellas. A pesar de que en sus orígenes estuvo ligado a JavaScript y su notación se basó en la notación de objetos JavaScript, JSON se ha convertido en un estándar independiente de datos que no está ligado a ningún lenguaje en concreto.

En el siguiente código se muestra un ejemplo de como sería el formato JSON en un mensaje de SAMP.

```
1 {"samp.mtype":"samp.hub.event.register",
2 "samp.params":{"id":"c2"}}
```

Capítulo 3

Estado del Arte

En este capítulo se repasarán las capacidades y funcionalidades que posee actualmente JSAMP, además de explicar como el Hub SAMP interactúa con otros aplicaciones, tales como Aladin y Topcat [24].

Cuando una aplicación se conecta a un Hub a través del protocolo SAMP, este comprueba que la conexión proviene de un host permitido. Como actualmente el sistema solo acepta conexiones locales, si la aplicación no se ejecuta en el mismo equipo que el Hub objetivo, se obtiene el error de la Figura 3.1.

Figura 3.1: Error al conectar aplicación no permitida.

Una vez hecha esta comprobación, si la aplicación se encuentra en el mismo equipo que el Hub, se le permite registrarse sin tener en cuenta el contenido de los metadatos. En la Figura 3.2 se muestran los mensajes que utiliza cualquier aplicación para registrarse contra el HUB y los metadatos (Figura 3.3) con los que queda registrado en ella.

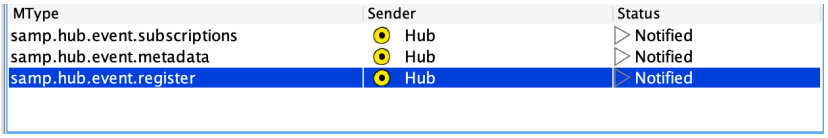


Figura 3.2: Mensajes de Registro de aplicaciones en HUB SAMP

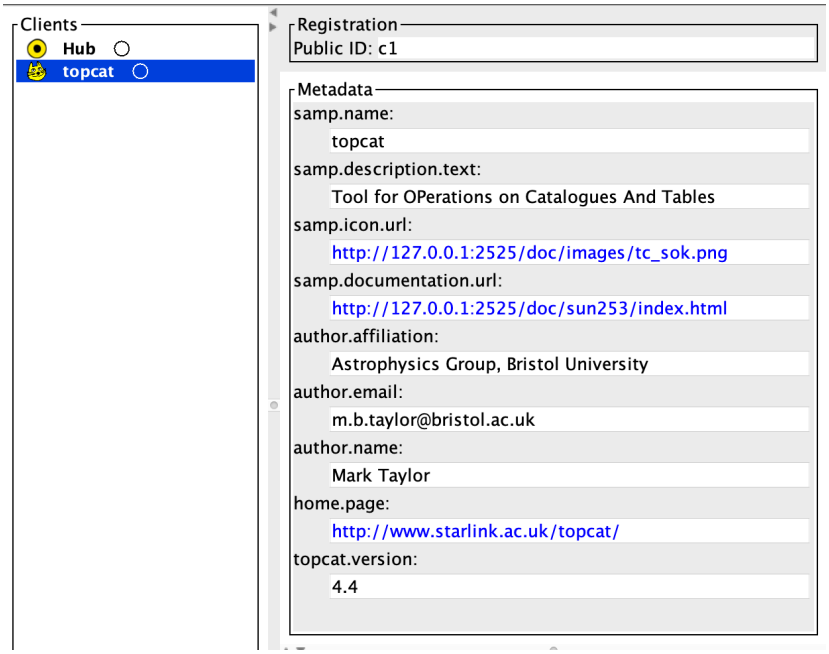


Figura 3.3: Formato metadatos SAMP

Una vez la conexión es aceptada, la aplicación entrante indica los tipos de datos a los que se suscribe y a partir de este momento está disponible para recibir cualquiera de los tipos a los que está suscrito (Figura 3.4).

Figura 3.4: Ejemplo de suscripciones de una aplicación en Hub Samp (Topcat)

En la versión actual de SAMP, las comunicaciones no se cifran y los mensajes van en texto claro, por lo que cualquier persona podría interceptar los mensajes si estos viajasen a través de una red. Los mensajes de SAMP, funcionan de manera que cuando un cliente envía un mensaje a otro, recibe otro mensaje de respuesta indicando el resultado y el estado del mensaje enviado. En la Figura 3.4 se muestra un mensaje de ping enviado desde el Hub al cliente Topcat.

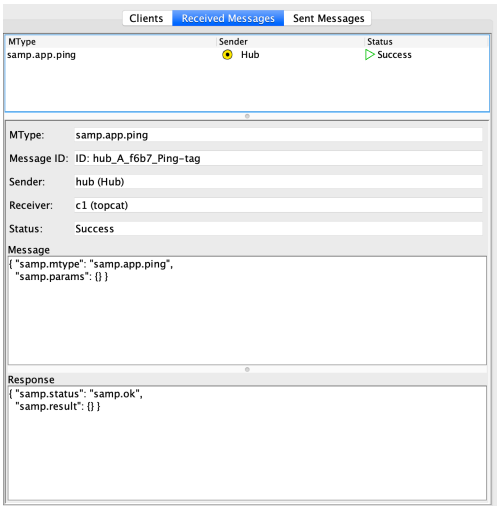


Figura 3.5: Ejemplo de mensaje de ping a Topcat

Para desconectar una aplicación del Hub, se envía un mensaje de desconexión y se eliminan del Hub todos los datos de la aplicación.

Funcionamiento de aplicaciones SAMP

Cuando dos aplicaciones se suscriben al mismo tipo de datos, se pueden comunicar entre si enviándose datos a través del Hub (Figura 3.4). A continuación se muestra un ejemplo del funcionamiento de Topcat y Aladin estando conectados al mismo Hub SAMP.

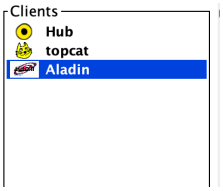
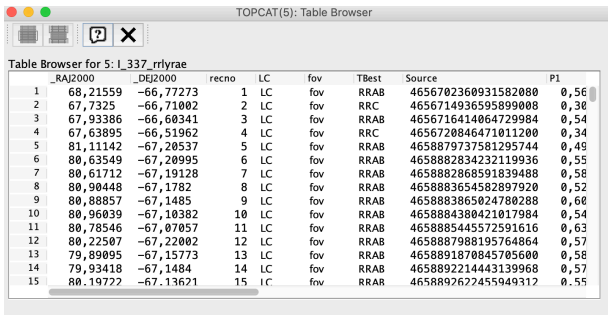


Figura 3.6: Topcat y Aladin conectados al Hub SAMP

1. Se cargan tablas con contenido astronómico en Topcat (Figura 3.7). Topcat permite cargar tablas a partir de archivos o seleccionar servidores desde los que podemos descargar catálogos guardados para poder utilizar sus tablas. Una vez seleccionado un catálogo, se cargan las diferentes tablas disponibles. Estas tablas almacenan en columnas los diferentes datos de objetos celestes que determinan su posición y propiedades.



	_RAJ2000	_DEJ2000	recno	LC	fov	TBest	Source	P1
1	68,21559	−66,77273	1	LC	fov	RRAB	4656702360931582080	0,56
2	67,7325	−66,71002	2	LC	fov	RRC	4656714936595899008	0,30
3	67,93386	−66,60341	3	LC	fov	RRAB	4656716414064729984	0,54
4	67,63895	−66,51962	4	LC	fov	RRC	4656720846471011200	0,34
5	81,11142	−67,20537	5	LC	fov	RRAB	4658879737581295744	0,49
6	80,63549	−67,20995	6	LC	fov	RRAB	4658882834232119936	0,55
7	80,61712	−67,19128	7	LC	fov	RRAB	4658882868591839488	0,58
8	80,90448	−67,1782	8	LC	fov	RRAB	4658883654582897920	0,52
9	80,88857	−67,1485	9	LC	fov	RRAB	4658883865024780288	0,60
10	80,96039	−67,10382	10	LC	fov	RRAB	4658884380421017984	0,54
11	80,78546	−67,07057	11	LC	fov	RRAB	4658885445572591616	0,63
12	80,22507	−67,22002	12	LC	fov	RRAB	4658887988195764864	0,57
13	79,89095	−67,15773	13	LC	fov	RRAB	4658891870845705600	0,58
14	79,93418	−67,1484	14	LC	fov	RRAB	4658892214443139968	0,57
15	80,19722	−67,13621	15	LC	fov	RRAB	4658892622455949312	0,55

Figura 3.7: Tablas del DR1 de la misión Gaia

2. Se carga una imagen de base en Aladin sobre la que mostrarán los datos de las tablas importadas (Figura 3.8). Aladin permite la carga de mapas sobre los que se dibujarán las estrellas. Se trata de una imagen fija con forma de esfera que se usará para representar los datos de las tablas.

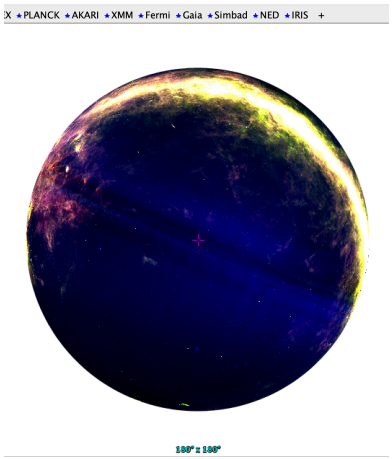


Figura 3.8: Vista de Iris en Aladin sin datos cargados

3. Se envían los datos de una tabla desde Topcat a Aladin (Figura 3.9). Desde Topcat, y tras seleccionar una de las tablas del catálogo, se pueden enviar a los diferentes clientes suscritos a este tipo de datos. En este caso se envían a Aladin.

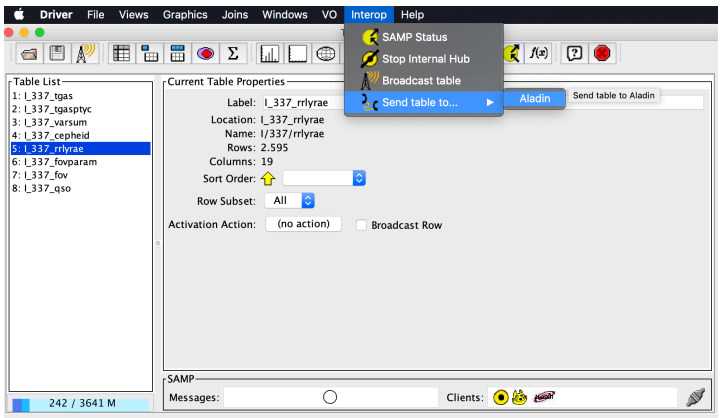


Figura 3.9: Envío de tablas desde Topcat a Aladin

Para que este envío de datos sea posible ambas herramientas han de estar suscritas al mismo MType. En el caso de este ejemplo ambas herramientas están suscritas al MType que aparece en la Figura 3.10 con el nombre de ”table.load.votable”.

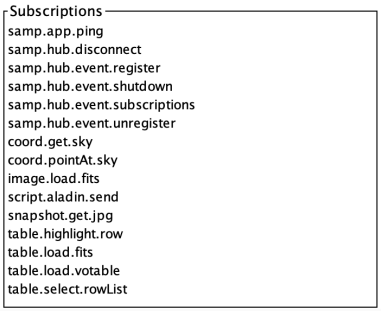
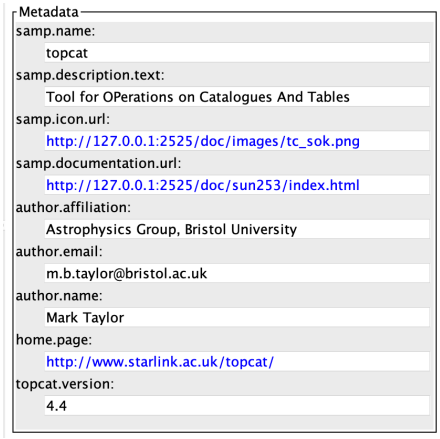


Figura 3.10: MTypes a los que se suscribe Aladin.

Para el envío de los mensajes se utilizan los metadatos(Figura 3.11) de cada una de las herramientas, con el fin de dirigirlos hace un lado u otro.



(a) Metadatos de Topcat.

(b) Metadatos de Aladin.

Figura 3.11: Metadatos de herramientas.

En el contenido del mensaje(Figura 3.12) de conexión entre las herramientas se indica el origen, destino y MType que se pretende enviar.

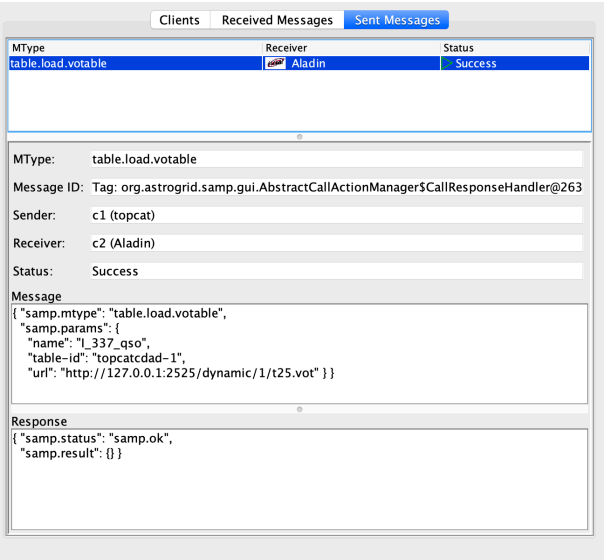


Figura 3.12: Mensaje entre Topcat y Aladin.

4. Vemos en el mapa de Aladin los datos de las tablas (Figura 3.13). En Aladin se representan sobre el mapa que hayamos seleccionado las estrellas y objetos celestes.

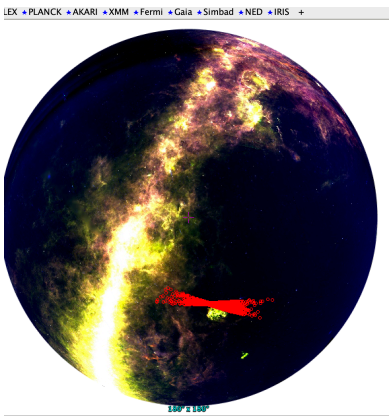


Figura 3.13: Datos de tabla de la DR1 de Gaia sobre Aladin

Capítulo 4

Estudio de viabilidad

En este capítulo trataremos la metodología utilizada para este proyecto, así como los tiempos utilizados para su realización y como ha sido su evolución en las diferentes etapas.

4.1 Metodología

El método de trabajo comprenderá una primera fase de estudio de las tecnologías empleadas para posteriormente realizar un proceso de desarrollo software mediante prototipado. Proporcionando en cada ciclo un prototipo del producto final, para comprobar su funcionamiento y estudiar la necesidad de nuevas funciones.

4.1.1 Estudio de tecnologías

Durante esta fase se realiza un estudio de todo el proyecto Gaia llevado a cabo por la ESA y sobre todo, centrándonos en el funcionamiento de SAMP. Además se realizan pruebas con el JSAMP actual para observar las características presentes y que podemos nosotros mejorar en este proyecto. Además de esto, se realiza un estudio del material inicial del que disponemos, que se trata de un proyecto Maven en java escrito por Mark Taylor.

4.1.2 Prototipado

La creación de prototipos consiste en la creación de un sistema de prueba que podrá ser evaluado por los usuarios finales. El propósito de los prototipos es el de servir de modelo preliminar sobre el que se pueden ampliar los requisitos y refinar las funciones ya existentes. Una vez los usuarios avalen el prototipo, éste puede servir de plantilla para el sistema definitivo.

Este método comienza con la recolección de requisitos, para ello tanto el desarrollador como el cliente definen unos requisitos principales. Sobre estos requisitos comienza en diseño y desarrollo. Las etapas del prototipado son las que aparecen en la figura 4.1.

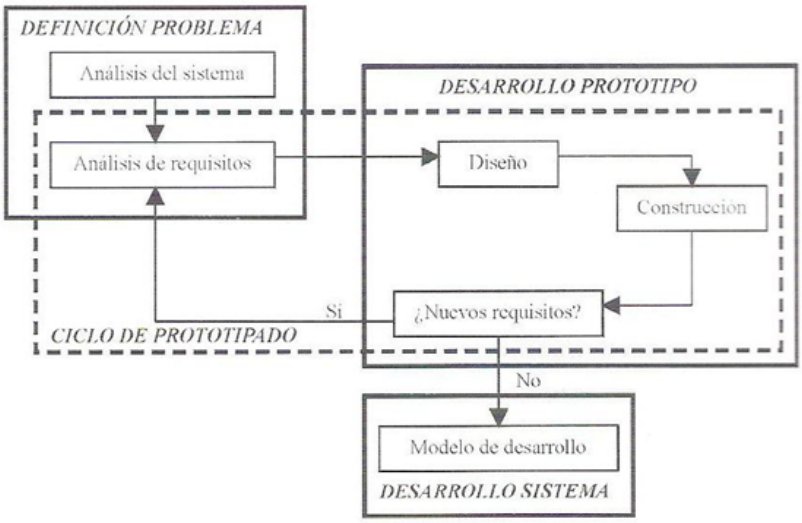


Figura 4.1: Ciclo de vida de prototipado

Analisis

Una vez realizada la primera recolección de requisitos, se realiza análisis de los requisitos y la especificación del primer prototipo, configurando el diseño básico del prototipo. El diseño del prototipo, lleva a la construcción del prototipo.

Construcción del prototipo

En la etapa de construcción, se construye una versión preliminar del producto, basándonos en los requisitos iniciales proporcionados por el cliente. En esta etapa debemos cumplir todos

los requisitos que nos haya propuesto el cliente, de la mejor manera posible, para a partir de este prototipo, en las fases siguientes poder definir nuevos requisitos.

Evaluación

El usuario realiza una prueba exhaustiva del prototipo, comprobando si se cumplen los requisitos establecidos en la fase inicial. En caso de que sea necesario, establecer nuevos requisitos o solicitar cambios en las funciones ya existentes.

Tras esto, el ciclo comienza de nuevo.

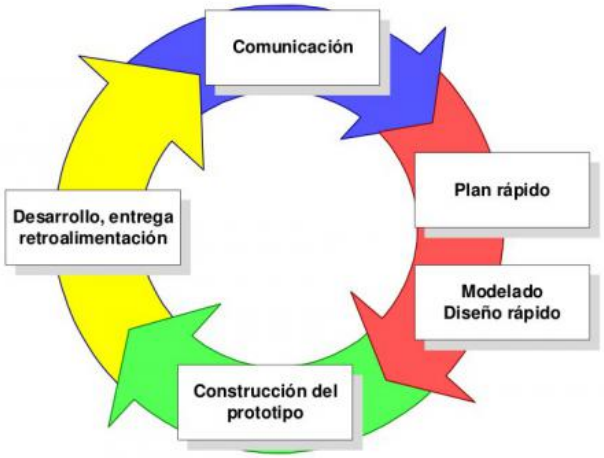


Figura 4.2: Prototipado

En caso de que el prototipo cumpla con todos las necesidades del usuario. Se pasa a una fase final del prototipado

Entrega y mantenimiento

Se entrega al usuario el sistema final, con la documentación necesaria sobre su uso y su mantenimiento.

4.2 Planificación

La realización de este proyecto comienza en Octubre de 2017 y tiene como fecha de finalización septiembre de 2019. Durante el periodo de realizacion de este proyecto, se ha compaginado con un trabajo a tiempo completo.

Se trabaja de lunes a jueves a partir de las 8 de la tarde, que es cuando finaliza la jornada laboral. Entre octubre y noviembre de 2017 se trabaja durante 2 horas diarias, ya que la carga de trabajo en la empresa es mínima y se puede avanzar en la realización del proyecto. A partir de noviembre el proyecto queda en pausa hasta agosto de 2018, en esta fecha se trabaja 4 semanas en el proyecto a razón de 4 horas diarias y el proyecto vuelve a quedarse en pausa debido al trabajo en la empresa. En noviembre se retoma durante 2 semanas el trabajo y posteriormente, en agosto de 2019 se retoma el proyecto hasta su finalización en septiembre.

Hay que tener en cuenta que la mayor parte del proceso consiste en el análisis del código existente, ya que sobre él se han implementado todas las funciones necesarias. A continuación veremos las fases en las que se ha dividido el proyecto, basándonos en el método de prototipado utilizado.

4.2.1 FASE 1: Establecer Comunicación y autenticacion LDAP

La principal ventaja que ofrece este protocolo es que es adecuado para el intercambio de datos tabulares, sobre todo en el ámbito de la astrofísica, lo que permite realizar estudios sobre los mismos datos en diferentes aplicaciones. Actualmente el Hub de JSAMP únicamente permite conexiones locales, por lo que el intercambio de datos se queda limitado a un mismo equipo, y por tanto una mejora de importancia, para que diferentes personas puedan trabajar desde diferentes equipos con los mismos datos, sería habilitar las conexiones desde otros equipos.. En esta fase el trabajo se ha centrado en investigar la mejor forma de habilitar este tipo de conexiones y analizar las consecuencias que tiene, por lo que se ha analizado el código Java, descubriendo que existe una función encargada de bloquear estas conexiones.

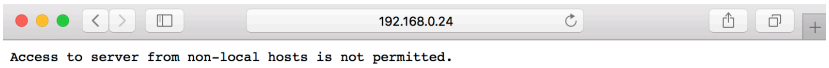


Figura 4.3: Hub bloqueado conexiones externas

Tras modificar estas secciones de código, el servidor comienza a aceptar conexiones entrantes desde el exterior. Esta situación no es muy segura, por lo que se implementa en un servidor externo un servidor LDAP, y en el JSAMP se implementa la necesidad de que toda conexión requiera un usuario, existente en ese LDAP, y una contraseña para establecer cualquier comunicación.

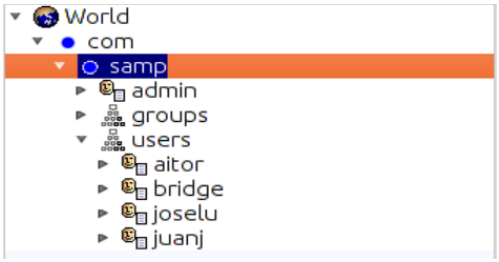


Figura 4.4: Estructura LDAP Utilizada

Con estas funcionalidades implementadas se realiza una revisión del primer prototipo.

Tras la revisión se establece la necesidad de que desde el HUB, exista la posibilidad de crear usuarios de LDAP, para el proceso de autenticación.

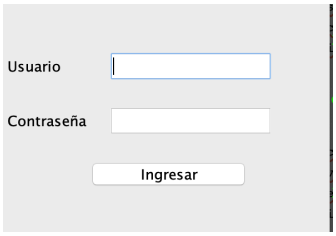
4.2.2 FASE 2: Cifrado de Comunicaciones

Para esta segunda fase se establece como objetivo que la comunicación entre el hub y los diferentes clientes vaya cifrada, ya que hasta este momento y con sistema actual, las comunicaciones son mediante HTTP y por lo tanto tanto las contraseñas como los datos viajan en texto plano, y podrían ser capturadas con cualquier software de análisis de tráfico de red.

Para realizar esto, se modifica el perfil web existente en la versión inicial de JSAMP, para que en lugar de utilizar HTTP, utilice conexiones cifradas mediante HTTPS, utilizando un certificado.

Además, como resultado de la fase anterior, se crea un menú superior en el HUB, que permita la creación de nuevos usuarios LDAP(figura 4.5).

(a) Menú Crear Usuario



(b) Ingresar Usuario

Figura 4.5: Proceso de creación de usuario LDAP

4.2.3 FASE 3: Retrocompatibilidad con versión estándar de SAMP

Tras la revisión del prototipo de la fase 2, se comienza con la fase 3, en la que se establece que el nuevo sistema JSAMP+, podrá conectarse con el sistema antiguo de SAMP. Esto se realizará a través de un Bridge, que establecerá una conexión puente entre los dos HUBs, conectándose a la versión antigua a través de HTTP y a la nueva mediante HTTPS.

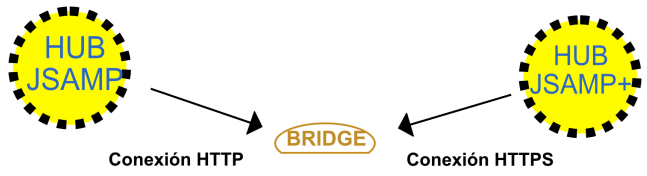
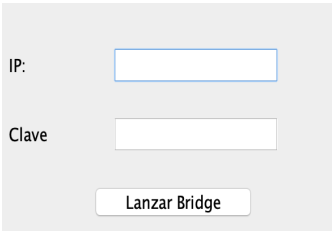


Figura 4.6: Funcionamiento de retrocompatibilidad mediante Bridge

Tras la revisión de esta fase se establece la necesidad de que el nuevo sofware se ejecute todo de una vez, por lo que al arrancar el nuevo JSAMP+, este arranque un bridge a un HUB JSAMP, y pueda aceptar conexiones tanto a la versión antigua como a la nueva.

Además, y tras una última revisión se implementa la opción de lanzar una conexión con otros hubs a traves de un menú, ya sean HUBs JSAMP o JSAMP+.(figura 4.7)

(a) Menú lanzar bridge



(b) Ingresar IP

Figura 4.7: Proceso creacion de Bridge

4.2.4 FASE 4: Detalles y pruebas finales

En esta fase final, se comprueba que todos los objetivos y los requisitos establecidos al inicio del proyecto se cumplen. Se realizan pruebas para determinar si el funcionamiento del programa es correcto y se cumplen con los estándares de calidad. Por último, se modifican pequeños detalles estéticos de la aplicación, que la diferencien de la original, para su entrega final.

4.3 Diagrama de Gantt

A continuación se muestra el diagrama de Gantt relativo a los tiempos reales de realización del proyecto.

Podemos observar las diferentes fases del modelo de prototipado. Una vez terminadas las 3 primeras fases del prototipado, tenemos un sistema completamente funcional que podemos presentar al cliente, aunque debido a pequeños detalles seguiremos con una 4ª fase para perfeccionar y darle una forma final al proyecto.

Además, al final de cada una de las fases, se realizan las pruebas necesarias para comprobar el correcto funcionamiento del programa.

Se observan grandes periodos sin actividad a lo largo de todo el proyecto debido a que se decidió dejar el proyecto, por falta de tiempo, por culpa del horario laboral y por problemas personales.

Hay que tener en cuenta que el recurso disponible para la realización de este proyecto es solo 1 persona, con los roles de analista, diseñador, y programador, por lo que la planificación del proyecto cambiará en el calculo de los costes.

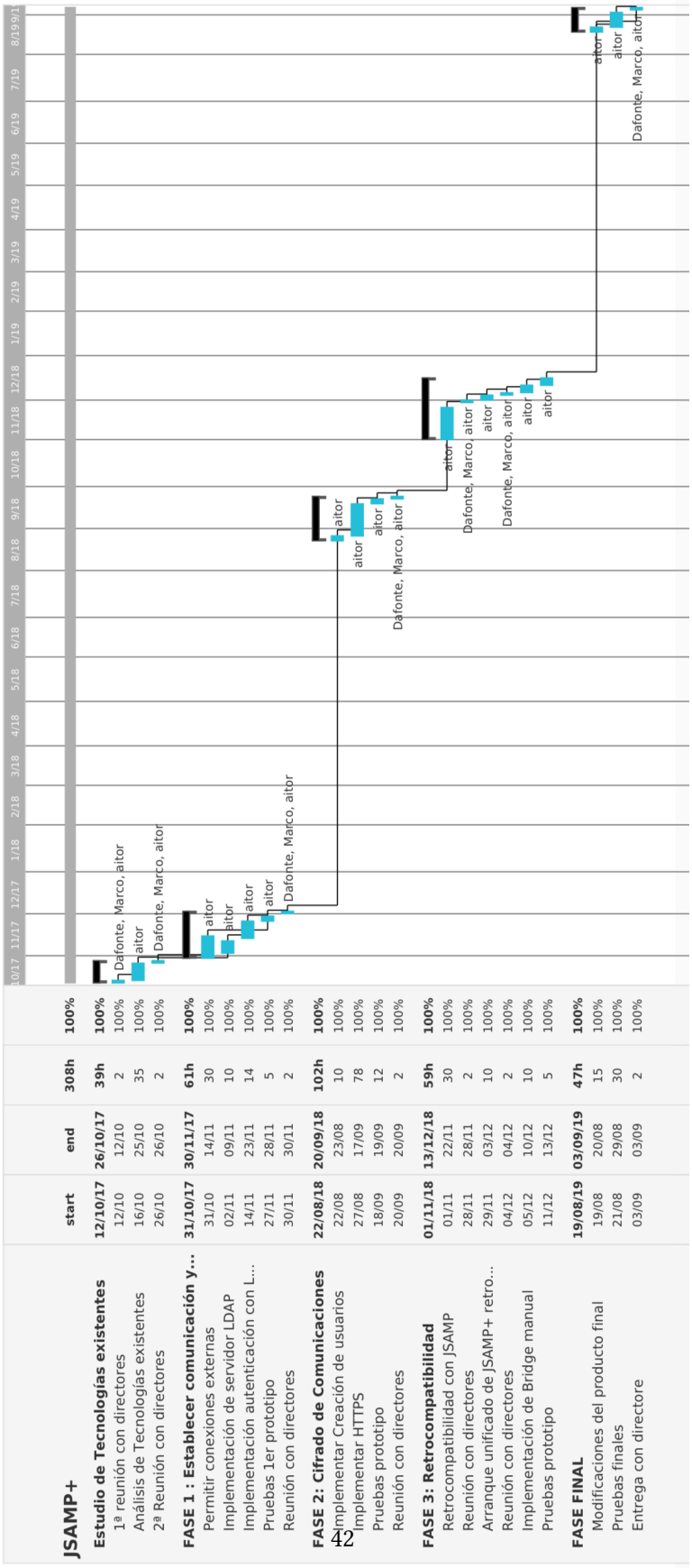


Figura 4.8: Diagrama de GANT del proyecto

4.4 Análisis de riesgos

En esta sección se localizarán los principales riesgos asociados al proyecto.

- R1: Diseñar e implementar funcionalidades de las que se desconoce su impacto real sobre la aplicación. Por ejemplo, no se sabe si la implementación del cifrado HTTPS, será capaz de convivir y ser compatible con la versión antigua de JSAMP.

- R2: Dificultad en el manejo de las tecnologías. SAMP es completamente nuevo para el programador. También es completamente desconocida la implementación utilizada en JSAMP.

- R3: Validación del proyecto: para la completa validación del proyecto, sería necesario que los desarrolladores de los clientes SAMP (topcat,aladin...) implementasen sus propios clientes en HTTPS, y con autenticación.

Una vez identificados, se realiza la clasificación de estos riesgos, para ello nos basaremos en su probabilidad de suceso y el impacto del mismo.

- **Probabilidad:**

  Probabilidad de que el riesgo se convierta en un problema. Puede ser Alta, Media o Baja.

- **Impacto:**

  Es el impacto que causa en la planificación, costes, tiempo... Puede ser Grave, Muy Grave o leve.

La siguiente tabla muestra una clasificación de los riesgos:

Probabilidad/Impacto	Muy Grave	Grave	Leve
Alta			R3
Media		R2	
Baja	R1		

Tabla 4.1: Riesgos Asociados al proyecto

4.5 Análisis de costes

En este punto se exponen los costes asociados al proyecto.

El personal para este proyecto lo constituyen 2 directories y una única persona.

Podemos clasificar el proyecto en 3 etapas respecto a la dedicación al mismo. En la primera etapa se trabajó durante unas 5 horas al día, desde el 12 de Octubre de 2017 al 31 de Noviembre del mismo año. Después se dejó un periodo de inactividad hasta la siguiente fase. Esta segunda fase comienza el 22 de Agosto, debido a vacaciones en el trabajo, al 20 de septiembre y el trabajo medio es de 4 horas al día.

En noviembre volvió a retomarse el proyecto y su dedicación media fue de 2h al día.

En la siguiente tabla se exponen los perfiles necesarios para la ejecución del proyecto y su coste por horas trabajadas:

Recurso	Coste(€/hora)
Analista	60
Diseñador	50
Tester	40
Desarrollador	40
Director	60

Tabla 4.2: Coste por puesto

Para calcular los costes totales del proyecto se han dividido todas las horas según el recurso que se utilizaría en cada fase, a pesar de que en este caso, todos los papeles han sido asumidos por una misma persona.

Etapa	Duración(horas)	Coste(€)
Estudio de Tecnologías	39	2.650€
Establecer comunciación y autenticación	61	2.720€
Cifrado de comunicaciones	102	4.390€
Retrocompatibilidad y Bridge	59	2.600€
Fase final	22	2.310€
TOTAL	283	12.360€

Tabla 4.3: Coste del proyecto

Como podemos observar el coste de nuestro proyecto tras 288 horas de trabajo ha sido de 12.360€.

Capítulo 5

Desarrollo

Tras una primera fase de documentación acerca de la tecnología SAMP y del estado inicial de JSAMP, se comienza con el desarrollo del proyecto. A lo largo de este capítulo se explican los pasos que se han seguido hasta obtener el producto final.

Como se ha explicado anteriormente, las diferentes funciones que se han ido implementando se encuentran en diferentes ciclos de prototipado, por lo que dividiremos este capítulo en las diferentes fases, para explicar en cada una de ellas, los pasos que se han seguido.

5.1 FASE 1: Establecer comunicación y autenticación LDAP

En esta primera fase se realiza un estudio del código disponible y las diferentes opciones y problemas que pueden surgir para cada una de funciones que se van a implementar.

5.1.1 Análisis

En este punto, tenemos un conocimiento general del proyecto Gaia, y sobre todo del protocolo SAMP y su funcionamiento.

En este paso se realiza un estudio del proyecto JSAMP, realizado por Mark Taylor. Se trata de un proyecto MAVEN, realizado en JAVA, con la estructura de la Figura 5.1:

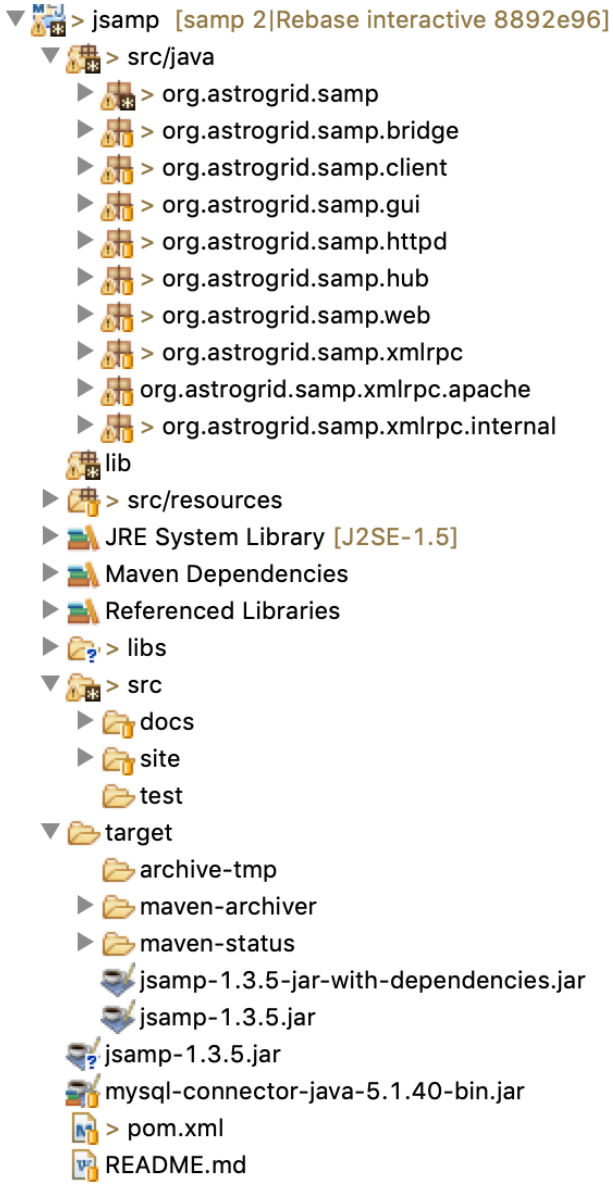


Figura 5.1: Estructura del proyecto JSAMP

En los diferentes paquetes que vemos, se observan las diferentes funciones.

- SAMP: En este paquete se definen los clientes, los tipos de mensajes, la interfaz, suscripciones, y los metadatos que se incluyen en cada mensaje, entre otras cosas.
- Bridge: Como su nombre indica se establece la clase Bridge y los proxys que se utilizan para conectar los bridges.
- Client: En el paquete Client se establecen las respuestas, conectores y como maneja el Hub los diferentes mensajes enviados entre los clientes.
- GUI: Se define la interfaz gráfica de JSAMP.
- HUB: Establece el funcionamiento básico del HUB
- HTTPD: Establece los servidores que se utilizan para las conexiones.
- WEB y XMLRPC: Establece los diferentes perfiles existentes para tratar las conexiones, según sean clientes o bridges los que realizan la conexión.

Una vez llega una conexión, se crea un hilo de ejecución de un HTTPServer, a través del cual se emitirá una respuesta. Esta conexión se analiza y en caso de que la dirección de origen sea externa, es bloqueada.

Tras el análisis del código del proyecto se detecta que en la clase: CorsHttpServer, existe la función isPermittedHost, que solo permite las conexiones desde host locales:

```
     * @param   address  socket address
     * @return  true  iff address is known to be permitted
     */
    public boolean isPermittedHost( SocketAddress address ) {
        if ( address instanceof InetSocketAddress ) {
            InetAddress iAddress = ((InetSocketAddress)
address).getAddress();
            if ( iAddress == null ) {
                return false;
            }
            else if ( iAddress.isLoopbackAddress() ) {
                return true;
            }
            else if ( isExtraHost( iAddress ) ) {
                return true;
            }
            else {
                try {
                    return iAddress.equals(
InetAddress.getLocalHost() );
                }
                catch ( UnknownHostException e ) {
                    return false;
                }
            }
        }
        else {
            logger_.warning( "Socket address not from internet? " +
address );
            return false;
        }
    }
```

5.1.2 Diseño e implementación

En esta primera fase se ejecutarán los procesos de diseño e implementación, dado que para permitir la conexión externa no es necesario modificar el diseño del código del programa, pero el diseño de la autenticación LDAP, requiere de un proceso de diseño en el que se decide el programa y las clases a utilizar.

Para permitir las conexiones desde fuera del host local, se cambia la función para que siempre devuelta ”TRUE” y cualquier dirección sea un host permitido.

A partir de este punto cualquier conexión hacia el Hub será permitida, ya venga del host local o de una dirección externa, ante esto, y como ya hemos analizado anteriormente, se establece la necesidad de un proceso de autenticación.

Para la autenticación se decide que la mejor solución es la creación de un servidor LDAP. Los motivos por los cuales se decide implementar un servidor LDAP son:

- Gran facilidad de implementación.
- Se crea un directorio con los datos de todos los usuarios, que son fácilmente accesibles desde cualquier aplicación que utilice LDAP.

Creación del servidor LDAP

Para la creación del servidor LDAP se utilizará una máquina externa, para poder acceder a ella desde cualquier instancia lanzada en una red. En nuestro caso se crea una máquina virtual Ubuntu.

En esta máquina virtual se instala un servidor LDAP, en nuestro caso se decide la utilización de la plataforma JXplorer. JXplorer es una plataforma LDAP, que permite editar, crear y examinar un directorio LDAP standard.

Se trata de un cliente completamente funcional de LDAP muy fácil de utilizar que nos permite implementar sin dificultades un servidor de prueba de LDAP para nuestro proyecto. En la Figura 5.2 se muestra la aplicación JXplorer corriendo en nuestra máquina Ubuntu y la estructura LDAP creada.

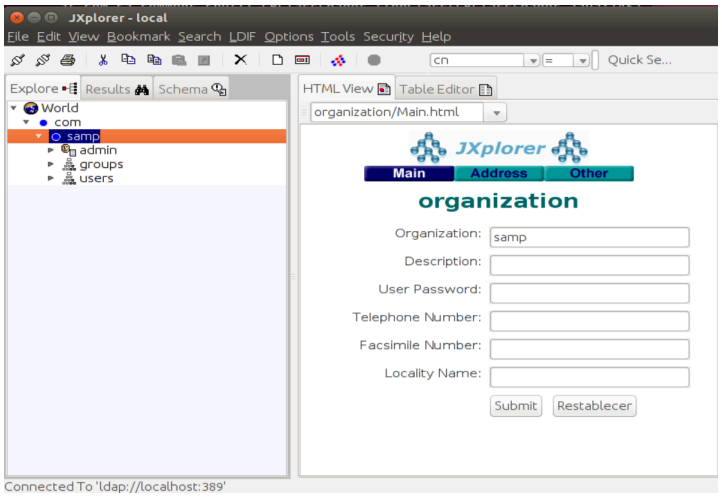


Figura 5.2: JXplorer

Implementación de la autenticación

Para la implementación de la autenticación contra el servidor de LDAP, se modificarán los metadatos que se deben incluir en cada conexión, para que se incluya un nombre de usuario y una contraseña.

Además, para la conexión contra el servidor LDAP, se crea una nueva clase dentro del paquete SAMP, que recibe el nombre LDAP, que será la encargada de gestionar la conexión. Para poder establecer dicha conexión también será necesario importar el paquete LDAPConnection.

```
.....
import com.novell.ldap.LDAPConnection;

public class ldap {
    private int ldapPort;
    private int ldapVersion;
    private LDAPConnection lc;
....
}
```

De esta manera, toda conexión que llegue al Hub de SAMP será analizada y si los metadatos no incluyen ningún usuario ni contraseña, la conexión será rechazada. En caso de que existan usuario y contraseña, se comprobará contra el servidor externo LDAP si el usuario existe y si la contraseña es correcta, una vez realizada esta comprobación, se establecerá la conexión.

5.1.3 Pruebas

Tras la realización de estos 2 pasos, no disponemos de aplicaciones para comprobar si las funciones que hemos implementado funcionan correctamente.. Este problema surge debido a que los clientes existentes (Aladin, Topcat) no utilizan usuario y contraseña para registrarse contra el Hub y además solo realizan Hub Discovery localmente.

Por lo tanto, para realizar esta prueba se utiliza un MONITORSAMP, escrito en HTML , modificado para que se conecte con HUBs externos y en sus metadatos envíe usuario y contraseña.

Este monitor HTML, no es más que un cliente que se conecta contra el Hub de JSAMP+ capaz de ver todos los clientes conectados y enviar mensajes de ping a los mismos.

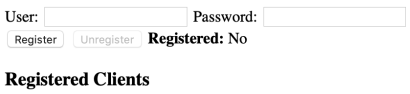


Figura 5.3: Monitor Web antes de conectar con el Hub.

En este Monitor, se introduce un usuario y contraseña antes de registrarse contra el Hub, en caso de que el usuario y contraseña sean correctos, se registra y solicita al hub los datos de los clientes registrados. Si el usuario y la contraseña son incorrectos salta un error.

Una vez registrados, muestra una lista con los clientes registrados, pudiendo desplegar una lista de los metadatos de cada uno de los clientes. Además, para cada uno de los clientes registrados, podemos lanzar mensajes de ping.

Para poder completar estas pruebas, sería necesario que las aplicaciones clientes incluyeran en sus metadatos un usuario y una contraseña.

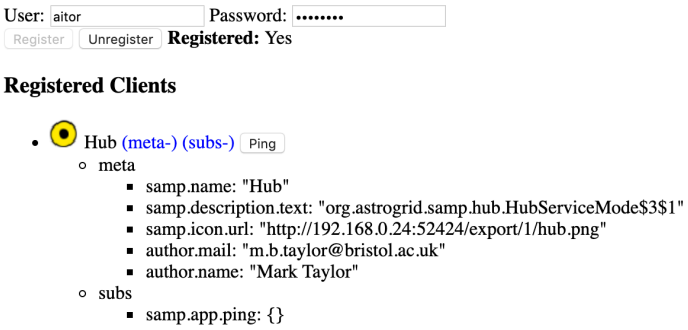
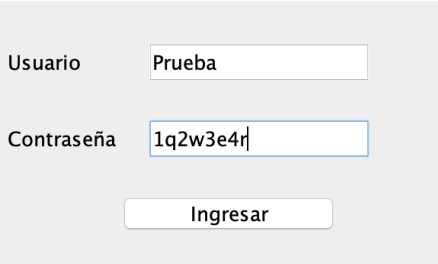


Figura 5.4: Monitor Web registrado.

5.2 FASE 2: Cifrado de Comunicaciones

Comenzamos esta segunda fase con la necesidad de establecer un menú para la creación de usuarios LDAP en el servidor remoto. Para ello, a través de la clase LDAP de la que hablamos en el punto anterior, se utiliza una función "ConexionManager" que, utilizando un usuario con permisos, permite crear usuarios en el servidor LDAP.



(a) Crear usuario Prueba.

(b) Usuario Prueba en LDAP.

Figura 5.5: Pruebas de creación de usuario.

Además, en la interfaz gráfica se crea un nuevo menú, capaz de crear usuarios, como

vemos en la Figura 4.5.

Para probar que esto funciona correctamente, creamos un nuevo usuario en el menú y comprobamos que se crea en el JXplorer (Figura 5.5). Posteriormente, probamos el login desde el monitor con este usuario recién creado.

5.2.1 Análisis

Para comenzar con el cifrado de las comunicaciones se realiza primero un estudio del funcionamiento actual de las conexiones del Hub y de los clientes. Observamos que todas las conexiones utilizan una clase HTTPServer. Para establecer las conexiones, este HTTPServer, utiliza la clase de Java, ServerSocket, y la clase Socket que utiliza conexiones HTTP.

```
1 .....
2 public class HttpServer {
3     private final ServerSocket serverSocket_;
4     private boolean isDaemon_;
5     private List handlerList_;
6 ......
```

Por lo tanto para esta nueva fase nos queda claro que necesitamos que las clases Server utilicen conexiones cifradas con SSL.

5.2.2 Implementación del Cifrado de comunicaciones

Para cifrar las comunicaciones comenzamos con la creación de una nueva clase HTTPS-Server, basándonos en la clase HTTPServer y modificando las diferentes funciones utilizadas. Esta nueva clase utilizará SSLServerSocket en lugar de la clase ServerSocket que utilizaba la versión anterior.

```
1 .....
2 public class HttpsServer {
3     private final SSLServerSocket serverSocket_;
4     private boolean isDaemon_;
5     private List handlerList_;
6     private final URL baseUrl_;
7 ......
```

Al no disponer de un certificado firmado, creamos certificados personales sin firma para usarlos en el cifrado y posteriormente creamos en el código de HTTPSServer la instrucción para que se salte la comprobación de la validez de este certificado (Figura 5.6).

Tras crear esta clase HTTPSServer, debemos modificar los perfiles y clases que utiliza SAMP para que las conexiones que utilice sean cifradas. Para ello se duplican todas clases que utilizan la clase HTTPServer y se modifican para que utilicen la clase HTTPServer, tras esto,

se renombran las clases antiguas a ”nombredeclaseHTTP.java.”

Figura 5.6: Certificado sin firmar para el ServerHTTPS.

En la Figura 5.7 vemos un ejemplo de las nuevas clases.

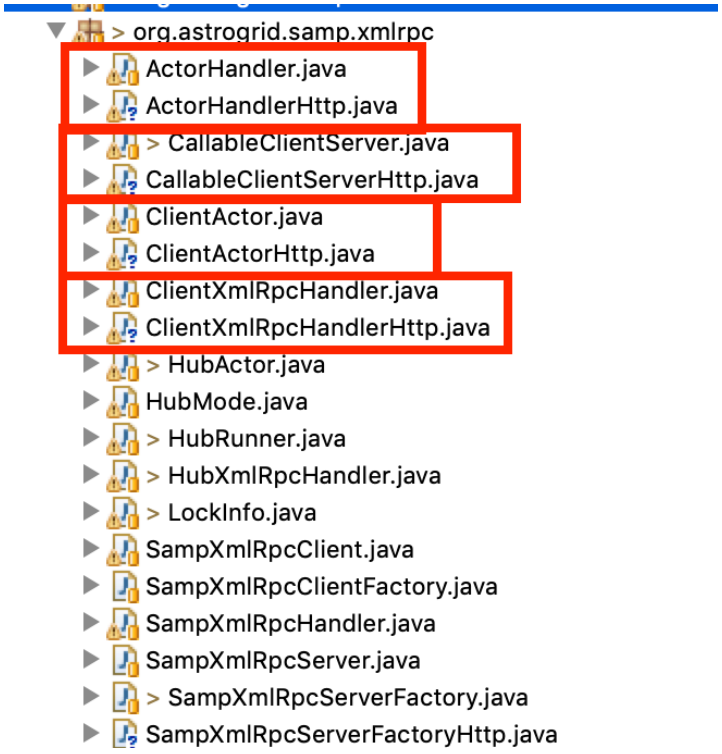


Figura 5.7: Ejemplo clases HTTP y HTTPS

5.2.3 Pruebas

Una vez creadas estas clases, el siguiente paso es realizar la prueba, para ello, modificamos el monitor para que se conecte mediante HTTPS al nuevo servidor. La primera vez que lo hemos probado no nos funciona y tras revisar el visor de eventos del explorador, vemos que no es capaz de comprobar la validez del certificado. Este problema es resultado de no disponer de un certificado válido, y para que el explorador nos permita realizar la petición contra el

servidor HTTPS, es necesario aceptar manualmente el certificado desde una nueva pestaña del mismo. Una vez aceptado el certificado, el MONITOR logra conectarse correctamente y es capaz de enviar pings al Hub HTTPS.

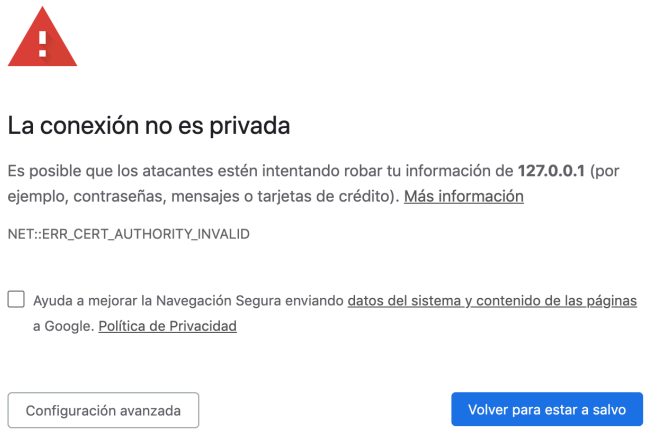


Figura 5.8: Error de certificado en Monitor.

Para comprobar el correcto funcionamiento general de esta funcionalidad, sería necesario modificar los clientes ya existentes para que, además de enviar un usuario y una contraseña, las comunicaciones que establezca vayan cifradas mediante SSL. Este es uno de los riesgos de los que hablamos anteriormente, ya que no tenemos acceso al código

5.3 FASE 3: Retrocompatibilidad con versión estándar de SAMP

Tras realizar un nuevo análisis de las funcionalidades que tenemos y ver que todo funciona correctamente hasta el momento, nos vemos en la necesidad de que esta nueva versión de SAMP sea compatible con versiones anteriores de SAMP.

5.3.1 Análisis

Tras examinar el código, llegamos a la conclusión de que la única manera de dar cabida a una compatibilidad de esta nueva versión que utiliza HTTPS con la versión anterior que utiliza HTTP, es a través de los bridge que utiliza la versión antigua para poder interconectar HUBS (Figura 5.9).

Antes de comenzar se realizan pruebas para comprobar el funcionamiento del bridge entre dos hubs JSAMP. El funcionamiento consiste en, teniendo dos hubs funcionando, lanzar una versión bridge de JSAMP pasando la URL del hub externo y una clave secreta compartida que se establece

a la hora de lanzar el Hub.

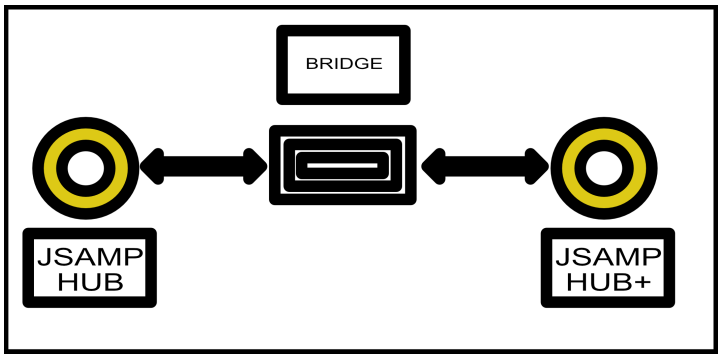


Figura 5.9: Funcionamiento Retrocompatibilidad

5.3.2 Diseño e implementación

Ya que el Hub clásico de SAMP usa la versión antigua de JSAMP, es necesario que la conexión del bridge hacia dicho hub use una conexión HTTP sin cifrar, en cambio la conexión contra el Hub JSAMP+ seguirá una conexión HTTPS cifrada por SSL igual que las conexiones de los clientes.

Para realizar esta conexión, el Hub origen necesita conocer una clave secreta y la URL de conexión del Hub destino. La clave es generada automáticamente una vez se lanza el Hub y es almacenada dentro de un fichero oculto ubicado en la carpeta HOME del sistema, para que solo SAMP pueda leerlo, junto con la URL del Servidor de destino (Figura 5.10).

```
sh-3.2# cat $HOME/.samp
# SAMP Standard Profile lockfile written Fri Aug 30 18:57:04 CEST 2019
# Note contact URL hostname may be configured using jsamp.localhost property
samp.secret=a193996703b6050f
samp.hub.xmlrpc.url=http://127.0.0.1:50687/xmlrpc
samp.profile.version=1.3
hub.impl=org.astrogrid.samp.hub.Hub$1
profile.impl=org.astrogrid.samp.xmlrpc.StandardHubProfile
profile.start.date=Fri Aug 30 18:57:04 CEST 2019
```

Figura 5.10: Archivo donde se almacenan las variables del HUB

El funcionamiento consiste en lanzar un hub JSAMP, posteriormente lanzar un Hub JSAMP+ y una vez los dos hubs están arrancados, se lanza una versión BRIDGE de JSAMP+, que se conectará primero al Hub seguro (JSAMP+) y posteriormente, usando las variables de entorno almacenadas en Figura 5.10, establecer la conexión contra el Hub clásico, y así posibilitar la comunicación entre clientes conectados a la versión antigua y clientes conectados a la nueva versión. Para lograr esta funcionalidad se modifica la clase Bridge, estableciendo que se conecte por un lado al nuevo cifrando la conexión mediante HTTPS, y por el otro lado al Hub clásico utilizando HTTP.

Una vez el bridge crea las conexiones, aparece en la nueva versión del Hub. Este bridge funciona a modo de proxy y a partir de ahora todas las conexiones cliente hacia el Hub antiguo aparecerán como conectadas a través de un proxy.

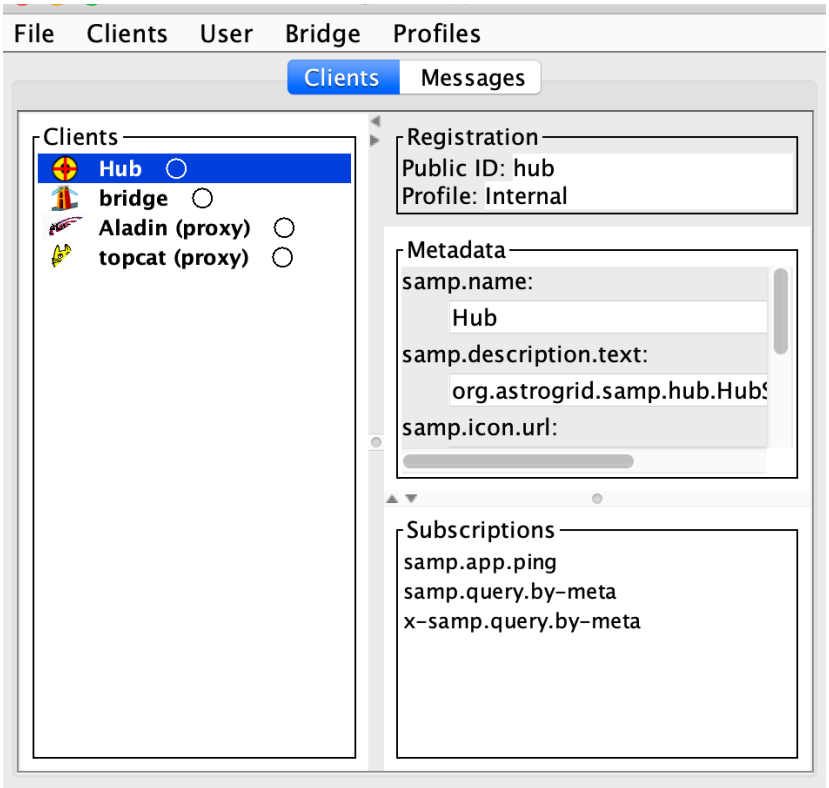


Figura 5.11: Clientes conectados al Hub JSAMP+ a través del Bridge.

Tras comprobar que funciona correctamente, se hace necesaria una manera de automatizar este proceso, y que al arrancar el hub de JSAMP+, directamente sea compatible con el hub clásico de JSAMP.

Para ello se crea un script, que arranca un Hub JSAMP sin interfaz gráfica y un Hub JSAMP+ normal, de manera que el usuario solo ve una interfaz y para él solo existe un HUB; posteriormente este script lanza el Bridge de conexión entre los dos HUBS.

```
#!/bin/bash
# -*- ENCODING: UTF-8 -*-

echo "JSAMP+ Pack Starting..."
java -jar jsamp-1.3.5-pro.jar hub -mode no-gui &
NUMBER=$!
echo $NUMBER
sleep 20
SECRET=$(sudo cat $HOME/.samp|grep samp.secret| cut -d '=' -f 2)
```

```
10 echo $SECRET
11 URL=$(sudo cat $HOME/.samp|grep samp.hub.xmlrpc.url|cut -d "=" -f 2)
12 echo $URL
13
14 java -jar jsamp-1.3.5-jar-with-dependencies.jar hub &
15 sleep 20
16
17 java -jar jsamp-1.3.5-jar-with-dependencies.jar bridge -standard
       -keys $URL $SECRET
18 echo "ESTA ES LA PASSWORD DEL BRIDGE"
19 echo $SECRET
20 echo "Esta es la url"
21 echo $URL
22
23 kill $NUMBER
24
25 echo "Adiós!"
```

5.3.3 Pruebas

Para poder comprobar que los objetivos de esta fase están cumplidos, se establece que la opción más adecuada consiste en la conexión de los clientes del estándar antiguo a través de la retrocompatibilidad mediante el Bridge y conectar el monitor al JSAMP+, para comprobar en éste que podemos ver los metadatos y las suscripciones de cada uno de los clientes.

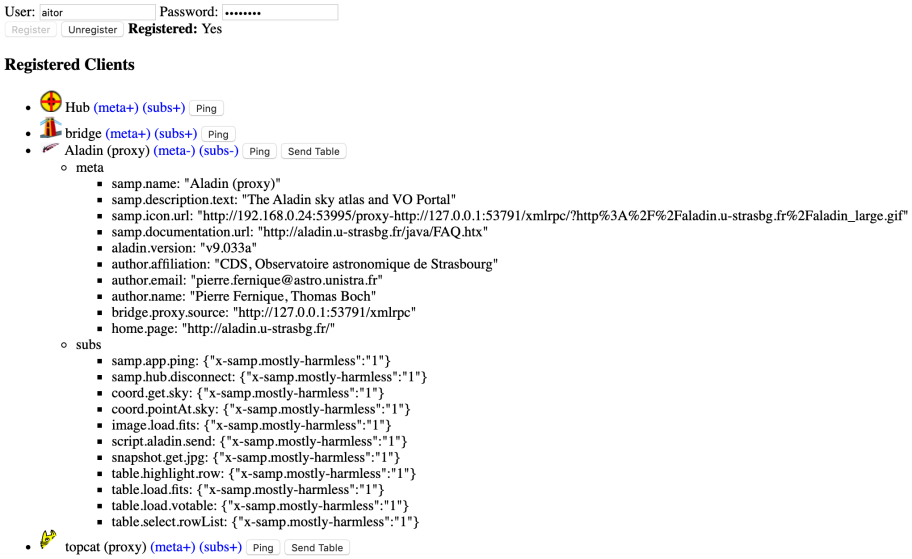


Figura 5.12: Retrocompatibilidad funcionado.

En primer lugar arrancamos el script y conectamos los clientes Topcat y Aladin para que

se conecten al HUB. Posteriormente arrancamos el monitor y establecemos la conexión con el JSAMP+. En la Figura 5.12 podemos ver los metadatos y las suscripciones de Aladin a través del Bridge, además vemos que nos da la opción para enviar una tabla.

Al pulsar enviar tabla se crea un mensaje que se envía a Aladin desde el monitor y que se puede ver desde el HUB JSAMP+. Como se observa en la Figura 5.13 la entrega del mensaje se ha completado con éxito. Este mensaje envía una serie de datos de prueba a Aladin, como podemos ver en la Figura 5.14.

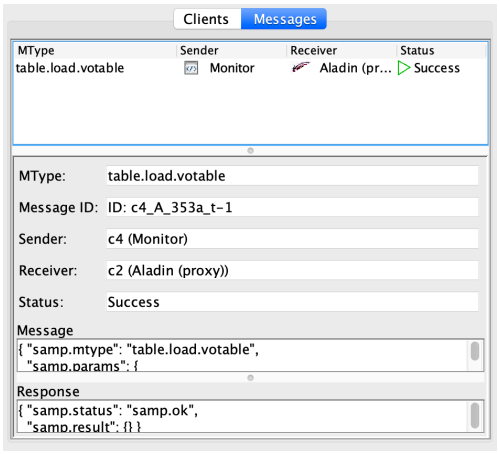


Figura 5.13: Mensaje de prueba entregado.

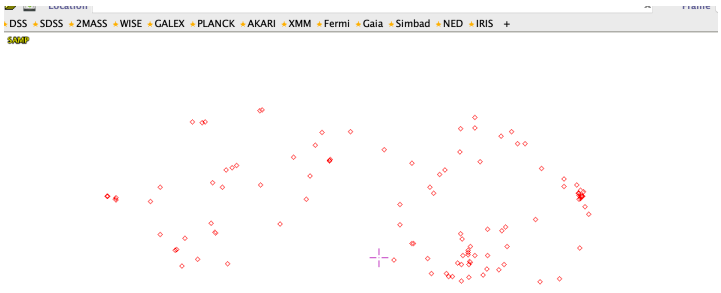


Figura 5.14: Datos de prueba en Aladin.

5.4 FASE 4: Detalles y pruebas finales

En esta fase y tras una última reunión con los directores se decide, que sería necesario poder probar la conexión bridge con otros HUBs. Además, se establecen diversos cambios para diferenciar esta versión de SAMP de la versión anterior.

Tras la revisión del prototipo anterior y la reunión con los directores del proyecto, se establecen nuevos requisitos y mejoras para el programa JSAMP+. Por un lado, se establece la necesidad de diferenciar esta nueva versión del la versión antigua de JSAMP mediante:

- Nuevo icono
- Nuevo nombre

Debido a la implementación de la retrocompatibilidad en la fase anterior se pierde la capacidad de operar desde aplicaciones en hosts diferentes, debido a que estas aplicaciones todavía utilizan el estándar clásico y su Hub no acepta conexiones de hosts remotos. A causa de esta pérdida de la capacidad de cooperación, surge la necesidad de poder interconectar Hubs JSAMP+ remotos entre sí, y que a través de estas conexiones se comuniquen las herramientas clásicas.

5.4.1 Diseño e implementación

En primer lugar y para diferenciar esta versión de la anterior, analizamos el código en busca de la clase que establece el icono que va a usar el Hub. Detectamos que la clase GuiHubService.java, establece el icono mediante la función:

```
.....
public class GuiHubService extends BasicHubService {
.....
    public JFrame createHubWindow() {
        JFrame frame = new JFrame( "SAMP+ Hub" );
        frame.getContentPane().add( createHubPanel() );
        frame.setIconImage( new ImageIcon( Client.class
                                           .getResource(
    "images/hub.png" ) )
                            .getImage() );
        frame.pack();
        return frame;
    }
.....
```

Por tanto decidimos modificar la imagen hub.png y crear un nuevo icono para la aplicación:

Figura 5.15: Nuevo Icono del HUB

Además en esa misma clase se cambia el nombre del JFrame de ”SAMP Hub” a ”SAMP+ HUB”.

Para volver a permitir la cooperación entre diferentes hosts, se crea un nuevo menú (Figura 5.16) en cual se puede establecer manualmente un Bridge con otros Hubs sin importar que estándar utilicen, si el nuevo o el clásico. En este menú se solicita una URL, en formato ”http://ip:puerto” o ”http://ip:puerto”, y la clave compartida para establecer la conexión con el servidor HTTP o HTTPS del Hub del otro extremo (Figura 5.17).

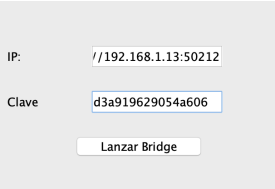


Figura 5.16: Nueva opción de menú para crear un Bridge.

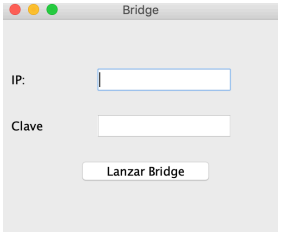


Figura 5.17: Datos necesarios para establecer el Bridge.

Una vez la conexión está establecida, los clientes de un lado pueden cooperar con los del otro (Figura 5.18). Como vemos en la imagen se crea un nuevo Bridge (2º en la imagen), que trae consigo la conexión a otro Bridge (3º en la imagen) ya que se lanzó otra instancia del JSAMP+ en otro equipo.

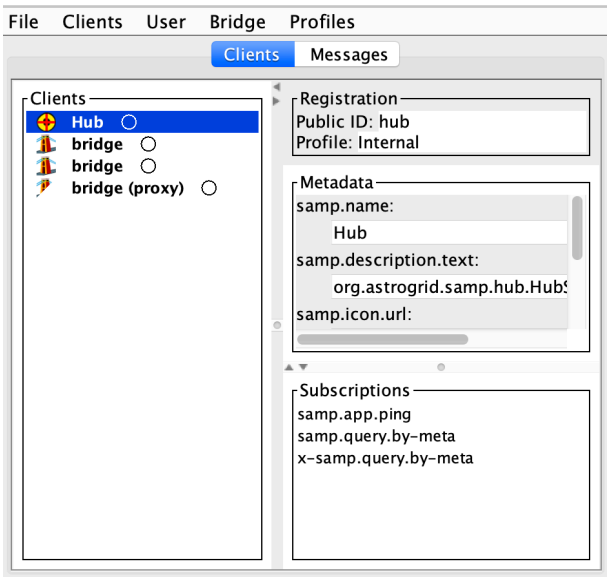


Figura 5.18: Bridge establecido en el JSAMP+.

5.4.2 Pruebas

Una vez llegado a este punto del proyecto, hay que comprobar que todas las funcionalidades que se establecieron en los objetivos del proyecto funcionan y que cumplen con un estándar adecuado. Para esto se diseña un experimento en el cual se lanzan en dos máquinas diferentes de la misma red, dos instancias del Hub de JSAMP+. Posteriormente se establece un Bridge entre estas dos instancias, para poder enviar datos de uno de los extremos al otro. A continuación se lanza un cliente en un lado que pueda cargar datos y otro cliente en el otro extremo que pueda recibir esos datos. Para finalizar la prueba se envían los datos desde un cliente al otro, lo que confirmará que todos los objetivos se han cumplido.

Para este experimento utilizaremos una máquina virtual y nuestro propio equipo. Una vez establecido el bridge entre ambas, lanzamos el Monitor de JSAMP+ en un extremo y Aladin SAMP en el otro, como se muestra en la Figura 5.19.

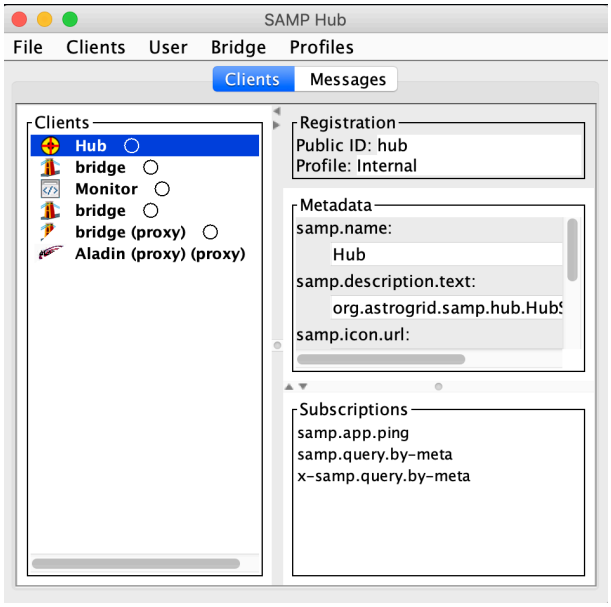


Figura 5.19: Hubs conectados con Aladin y Monitor.

Una vez establecida la conexión, se envía la tabla de datos de prueba al cliente Aladin remoto a través del Bridge establecido con la otra instancia de JSAMP+.

En la Figura 5.21 se muestra el resultado de enviar estos datos en el mapa de Aladin.

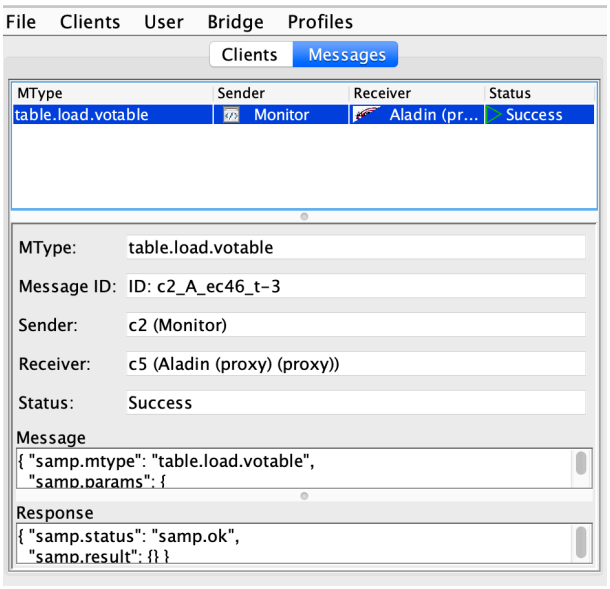


Figura 5.20: Mensaje de los datos enviados.

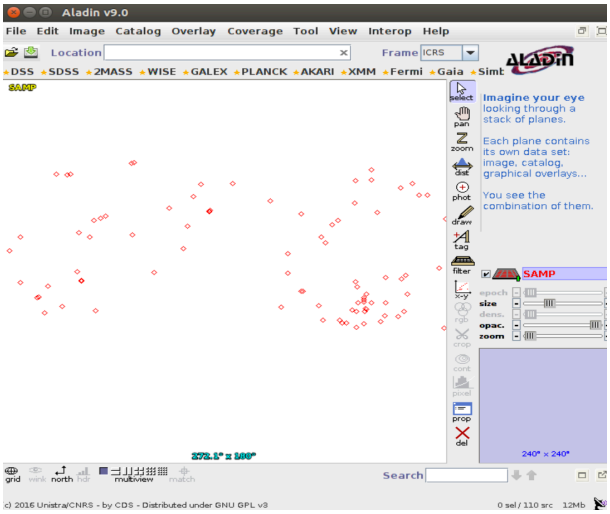


Figura 5.21: Datos sobre mapa de Aladin remoto.

Con estas pruebas se confirma el correcto funcionamiento de la aplicación y que se han implementado todas las funciones requeridas y las necesidades que han ido surgiendo a lo largo del desarrollo.

Capítulo 6

Conclusiones

El objetivo principal de este proyecto fue desarrollar un nuevo estándar de comunicaciones, JSAMP+, que a partir de SAMP utilice conexiones seguras para compartir datos y cooperar con otros usuarios. Este objetivo principal se ha completado y, si es adoptado, supondrá un avance para las herramientas utilizadas por la comunidad científica y no necesariamente sólo en astronomía sino que podría extenderse a otros ámbitos

La utilización de un desarrollo basado en prototipos también ha favorecido al profundo conocimiento del protocolo y las funciones a nivel individual, así como a facilitar la identificación de problemas en las diferentes fases del desarrollo y determinar los riesgos asociados a las distintas funciones que se quieren implementar.

Las conclusiones que se obtienen tras la realización de este proyecto son las siguientes:

- En JSAMP+ se ha conseguido incrementar de manera importante la seguridad proporcionada por el antiguo estándar SAMP mediante mecanismos de cifrado de las comunicaciones. El cifrado las comunicaciones mediante HTTPS, permite que las conexiones con hosts externos, pertenecientes a Internet o a otras redes, sean seguras, de manera que personas no autorizadas no pueden acceder a datos privados.

- Se ha establecido un método de autenticación mediante LDAP para proporcionar otro nivel adicional de seguridad, que impide que usuarios no deseados, con acceso a las herramientas que utilizan SAMP, puedan conectarse al Hub y obtener datos para los que no tienen autorización. Del mismo modo se llega a la conclusión de que la instalación de un servidor LDAP en un equipo remoto proporciona un login centralizado para conexiones de varios usuarios desde distintas ubicaciones. Además, este mecanismo de autenticación hace posible que se puedan registrar los cambios que se realizan sobre los datos, indicando qué usuario es el que realiza cada cambio.

- JSAMP+, al contrario que SAMP, permite conexiones desde hosts externos hacia el Hub, lo que posibilita la cooperación entre usuarios de diferentes localizaciones. Esto hace

más fácil el desarrollo de nuevas herramientas y el trabajo con grandes volúmenes de datos, lo que es de gran ayuda en el caso de los proyectos de astronomía.

- El estándar SAMP actual sigue siendo necesario, dado que las aplicaciones existentes todavía se ejecutan sobre HTTP y no requieren un usuario y contraseña para registrarse contra un HUB. Por lo tanto, en el desarrollo de JSAMP+, se ha conseguido una retrocompatibilidad con el estándar clásico, para poder continuar trabajando al tiempo que las aplicaciones existentes se van adaptando al nuevo estándar.

El conocimiento adquirido desde el principio en este proyecto ha sido de gran ayuda para que el desarrollo del mismo mediante el uso de las tecnologías fuese más sencillo y la planificación de las diferentes fases fuese la adecuada. Entre la gran cantidad de tecnologías sobre las que se ha adquirido conocimiento en este proyecto se destacan el protocolo SAMP, y tecnologías como Java, Maven, JSON, LDAP, HTML y Javascript.

Capítulo 7

Líneas Futuras

En este apartado se exponen funciones y mejoras que se podrían abordar en futuras versiones de JSAMP+, que o se encontraban fuera del alcance del proyecto o han ido surgiendo a lo largo de su desarrollo:

- Añadir un mecanismo de certificado que otorgue al cifrado de las comunicaciones mayor confianza y por lo tanto mayor seguridad. La utilización de certificados firmados por entidades certificadoras sería una de las opciones a valorar.
- Establecer un registro en el que guardar los cambios realizados y los accesos a los datos. Con el fin de aumentar la seguridad y establecer un protocolo para impedir la pérdida o el borrado accidental de información, se podría crear un mecanismo de log por el cual, se guarden todos los registros posibles sobre cada interacción con los datos.
- Establecer un repositorio común de datos en el que se almacene toda la información existente. Este repositorio se conectaría al Hub permitiendo que las diferentes herramientas accedan a estos datos y los puedan modificar. Podría estudiarse la posibilidad de crear una nueva herramienta para gestionar este repositorio o desarrollar una nueva funcionalidad en el Hub que se encargue de ello.
- Crear un repositorio con el nuevo estándar JSAMP+ y promocionarlo dentro de la comunidad astronómica. El propósito de este paso sería darle publicidad al nuevo estándar con el objetivo de que se asiente como el estándar básico para comunicaciones y trabajo con catálogos de datos,y extender su uso en diferentes ámbios, no sólo en el de la astrofísica.

Lista de acrónimos

SAMP *Simple Application Message Protocol.*

ESA *European Space Agency.*

IVOA *International Virtual Observatory Alliance.*

LDAP *Lightweight Directory Access Protocol.*

SSL *Secure Socket Layer.*

TLS *Transport Layer Security.*

HTTPS *Hyper-Text Transfer Protocol Secure.*

HTTP *Hyper-Text Transfer Protocol.*

DPAC *Data Processing and Analisys Consortium.*

IT *Information Technologies.*

CU *Coordination Unity.*

DPC *Data Processing Centers.*

DR *Data Release.*

EDR *Early Data Release.*

RPC *Remote Procedure Call.*

VO *Virtual Observatory.*

JSON *JavaScript Object Notation.*

TCP *Transmission Control Protocol.*

CSR *Certificate Signing Request.*

XML *Extensible Markup Language.*

Glosario

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases.

Órbita Lissajous es una órbita con trayectoria casi periódica que un objeto puede seguir alrededor de un punto de Lagrange dentro de un sistema de tres puntos sin requerir de un sistema de propulsión.

Cuásar es un objeto celeste que emite grandes cantidades de energía, con radiaciones similares a las de las estrellas. Los cuásares son centenares de miles de millones de veces más brillantes que las estrellas. Se piensa que pueden ser agujeros negros que emiten intensa radiación cuando capturan estrellas.

Framework es una estructura real o conceptual destinada a servir como soporte o guía para la creación de algo que convierta la estructura en algo de utilidad.

Application Discovery es un proceso a través del cual las aplicaciones instaladas y usadas en una empresa o proyecto son identificadas y recogidas.

Phising es un tipo de ataque de ingeniería social en el que se pretende robar información a usuarios, incluyendo credenciales, tarjetas de crédito o datos personales. Consiste en que el atacante se haga pasar por una entidad de confianza, para engañar a la víctima y que esta le entregue sus datos.

Ataques DOS es un ciberataque en el cual el atacante hace que una red o máquina no esté disponible para los usuarios a los que está destinada, a través del bloqueo temporal o indefinido de los servicios de un host.

Internet Engeneering Task Force (IETF) es una organización internacional abierta de normalización, que tiene como objetivos contribuir a la Ingeniería de Internet.

Request for Comments son una serie de publicaciones del grupo del IETF que describen diversos aspectos del funcionamiento de Internet y otras redes.

Maven es una poderosa herramienta de gestión de proyectos que se basa en POM (Modelo de Objetos de Proyecto). En breves términos, podemos decir que es una herramienta que se puede utilizar para construir y gestionar cualquier proyecto escrito en Java.

Bibliografía

[1] Samp, simple application messaging protocol. [En línea]. Disponible en: http://www.ivoa.net/documents/SAMP/20120411/REC-SAMP-1.3-20120411.html

[2] M. Taylor, T. Boch, and J. Taylor, "SAMP, the Simple Application Messaging Protocol: Letting applications talk to each other," *Astronomy and Computing*, vol. 11, no. PB, pp. 81–90, 2015.

[3] International virtual observatory association. [En línea]. Disponible en: http://www.ivoa.net/

[4] F. Bonnarel, P. Fernique, O. Bienaymé, D. Egret, F. Genova, M. Louys, F. Ochsenbein, M. Wenger, and J. G. Bartlett, "The ALADIN interactive sky atlas. A reference tool for identification of astronomical sources," *Astronomy & Astrophysics*, vol. 143, pp. 33–40, Apr. 2000.

[5] Aladin. [En línea]. Disponible en: http://aladin.u-strasbg.fr/aladin.gml

[6] Gaia Collaboration, T. Prusti, J. H. J. de Bruijne, A. G. A. Brown, A. Vallenari, C. Babusiaux, C. A. L. Bailer-Jones, U. Bastian, M. Biermann, D. W. Evans, and et al., "The Gaia mission," *Astronomy & Astrophysics*, vol. 595, p. A1, Nov. 2016.

[7] Proyecto espacial gaia. [En línea]. Disponible en: http://sci.esa.int/gaia/

[8] Agencia espacial europea. Fecha de acceso: Noviembre 2017. [En línea]. Disponible en: https://www.esa.int/ESA

[9] Java. [En línea]. Disponible en: https://docs.oracle.com/javase/specs/

[10] Jsamp. [En línea]. Disponible en: http://www.star.bristol.ac.uk/%7Embt/jsamp/

[11] Apache maven project. [En línea]. Disponible en: http://maven.apache.org/index.html

[12] Proyecto espacial hipparcos. [En línea]. Disponible en: http://sci.esa.int/hipparcos/

[13] Gaia's lissajous type orbit. Fecha de acceso: 19/06/2019. [En línea]. Disponible en: http://sci2.esa.int/interactive/media/flashes/5_5_1.htm

[14] Consorcio de procesamiento y análisis de datos. [En línea]. Disponible en: https://www.cosmos.esa.int/web/gaia/dpac/consortium

[15] Gaia Collaboration, A. G. A. Brown, A. Vallenari, T. Prusti, J. H. J. de Bruijne, F. Mignard, R. Drimmel, C. Babusiaux, C. A. L. Bailer-Jones, U. Bastian, and et al., "Gaia Data Release 1. Summary of the astrometric, photometric, and survey properties," *Astronomy & Astrophysics*, vol. 595, p. A2, Nov. 2016.

[16] Gaia Collaboration, A. G. A. Brown, A. Vallenari, T. Prusti, J. H. J. de Bruijne, C. Babusiaux, and C. A. L. Bailer-Jones, "Gaia Data Release 2. Summary of the contents and survey properties," *ArXiv e-prints*, Apr. 2018.

[17] Gaia edr3 announcement. [En línea]. Disponible en: https://www.cosmos.esa.int/web/gaia/news-2019#GaiaDR3Announcement

[18] Plastic - a protocol for desktop application interoperability. [En línea]. Disponible en: http://www.ivoa.net/documents/Notes/Plastic/PlasticDesktopInterop-20060601.html

[19] Rpc, remote procedure call protocol. [En línea]. Disponible en: https://www.ibm.com/support/knowledgecenter/en/ssw_aix_72/commprogramming/ch8_rpc.html

[20] Json. [En línea]. Disponible en: https://www.json.org

[21] P. Loshin, Ed., *Big Book of Lightweight Directory Access Protocol (Ldap) Rfcs*, 1st ed. Orlando, FL, USA: Academic Press, Inc., 2000.

[22] M. Martin-Romo, Ed., *Redes de Computadoras: Un Enfoque Descendente*, 5th ed. Madrid, Spain: Pearson Educación S.A., 2010.

[23] Java sockets. [En línea]. Disponible en: https://docs.oracle.com/javase/7/docs/api/java/net/Socket.html

[24] M. Taylor, "Topcat: Desktop exploration of tabular data for astronomy and beyond," vol. 4, no. 3, 2017. [En línea]. Disponible en: http://www.mdpi.com/2227-9709/4/3/18